~AGE~



Cahier des charges de l’EIP v 2.0

15/06/2014

César Leblic

Laureline Patoz

Anthony Boucher

Dorian Pinaud

Paul Baron

Philippe-Henry Sutter

Théo Bolognini

# 

# Résumé du document

Ce document présente la version 2.0 du cahier des charges de l’EIP AGE  (Another Game Engine). Il sert à donner une ligne de conduite à l’équipe de réalisation qui devra suivre son contenu, et il regroupe la liste des besoins, des exigences et des contraintes qu'il faudra respecter lors de la réalisation du projet.

Le projet est d'abord décrit dans son ensemble, puis point par point, les cibles visées sont ensuite citées, ainsi que les outils utilisés au cours du développement. La description des différentes parties du programme renseigne ensuite de manière plus approfondie le travail qui sera effectué. Pour finir, les différents outils qui permettront à l’équipe d’avoir une organisation optimale sont listés et l’aspect communication essentiel au projet est développé.

# Description du document

|  |  |
| --- | --- |
| Titre | CDC AGE |
| Date | 15 juin 2014 |
| Auteurs | César Leblic, Laureline Patoz, Anthony Boucher, Paul Baron, Dorian Pinaud, Philippe-Henry Sutter, Théo Bolognini |
| E-Mail | ageproject@googlegroups.com |
| Sujet | Cahier des charges pour le projet AGE |
| Version | 2.0 |

# Tableau des révisions

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Date | Version | Auteur | Section(s) | Commentaire |
| 22/04/14 | 1.0 | / | / | Version 1.0 du document |
| 01/06/14 | 1.1 | Dorian Pinaud | Résumé | Résumé du document |
| 01/06/14 | 1.2 | César Leblic | I.A | Précisions sur les partenaires du projet. |
| 01/06/14 | 1.3 | Laureline Patoz | III.G.1 | Précisions sur la méthode Scrum |
| 01/06/14 | 1.4 | Anthony Boucher | Annexes | Glossaire |
| 01/06/14 | 1.4 | Théo Bolognini | Annexes | Liens de retour du glossaire |
| 01/06/14 | 1.5 | Laureline Patoz | / | Corrections |
| 14/06/14 | 2.0 | Laureline Patoz | / | Mise en page |

Table des matières

[I. Présentation d’Epitech 1](#_Toc386195042)

[A. L’école 1](#_Toc386195043)

[B. Présentation de l’EIP 1](#_Toc386195044)

[II. Présentation d’AGE 1](#_Toc386195045)

[A. Présentation du produit 1](#_Toc386195046)

[B. Les objectifs techniques 2](#_Toc386195047)

[1. Moteur flexible et complet 2](#_Toc386195048)

[2. Performances 2](#_Toc386195049)

[3. Graphismes modernes 2](#_Toc386195050)

[C. Les objectifs sur le marché 3](#_Toc386195051)

[1. Immersion et réalité virtuelle 3](#_Toc386195052)

[2. Open Source 3](#_Toc386195053)

[D. Les cibles 3](#_Toc386195054)

[1. Utilisateurs potentiels 3](#_Toc386195055)

[2. Les profils techniques ciblés 4](#_Toc386195056)

[3. Les plateformes ciblées 4](#_Toc386195057)

[III. Présentation de l’environnement de réalisation 4](#_Toc386195058)

[A. Environnement de réalisation 4](#_Toc386195059)

[1. Langages et API utilisés 4](#_Toc386195060)

[2. Plateformes et outils de développement 5](#_Toc386195061)

[B. Environnement matériel 6](#_Toc386195062)

[1. Environnement de développement 6](#_Toc386195063)

[C. Architecture technique 6](#_Toc386195064)

[1. Le moteur de jeu 6](#_Toc386195065)

[2. Le moteur graphique 10](#_Toc386195066)

[3. Les outils d’éditions 14](#_Toc386195067)

[D. Composants existants 14](#_Toc386195068)

[1. Technologies intégrées au moteur 14](#_Toc386195069)

[2. Technologies de gestion de projet 16](#_Toc386195070)

[E. Stratégie de tests 16](#_Toc386195071)

[1. Tests unitaires 16](#_Toc386195072)

[2. Tests fonctionnels 17](#_Toc386195073)

[3. Tests d’acceptation 17](#_Toc386195074)

[4. Tests de performance 17](#_Toc386195075)

[F. Communication 17](#_Toc386195076)

[1. Identité visuelle 17](#_Toc386195077)

[2. Les réseaux sociaux 17](#_Toc386195078)

[3. Le site 18](#_Toc386195079)

[4. Wiki 18](#_Toc386195080)

[G. Organisation de projet 19](#_Toc386195081)

[1. Une méthode agile : Scrum 19](#_Toc386195082)

[2. Organisation de l’équipe en interne 19](#_Toc386195083)

[IV. Annexes 21](#_Toc386195084)

[A. Glossaire 21](#_Toc386195085)

# Présentation d’Epitech

Epitech est une école de formation d’experts en informatique et en nouvelles technologies. L’école, membre du groupe IONIS, est répartie sur 12 sites en France. Son objectif est de former des passionnés de l’informatique à l’aide d’une pédagogie innovante axée sur les besoins réels des entreprises et sur les modes d’apprentissage de la nouvelle génération.

Outre des projets obligatoires tout au long de leur formation, les étudiants créent un projet personnel de fin d’études : l’Epitech Innovative Project (EIP). Le choix du sujet est libre, à condition que le concept de base soit innovant et techniquement intéressant. Les étudiants forment alors un groupe composé d’au minimum six élèves de leur promotion afin de réaliser leur projet de la troisième à la cinquième année. La réussite de l’EIP sera évaluée sur leurs capacités à concevoir, réaliser et présenter leur projet. Tout au long de sa réalisation, le labEIP propose de nombreux suivis afin de guider les groupes dans leur progression, et leur permettant de mener un projet dans son intégralité, avec entre autres l'intégration d’une étude de l'existant, la mise en place d’un cahier des charges, la conception, la réalisation, la communication autour du projet, l’aspect marketing, etc. Ainsi, à son terme, le projet doit être fonctionnel et en mesure d’être utilisé dans le monde professionnel.

Le but de ce travail d’innovation est de fournir aux élèves d’Epitech une pratique de gestion de projet ainsi que l’expérience d’un travail en équipe. L’ensemble donne des bases solides aux étudiants pour leur entrée dans le monde professionnel, mais aussi des ouvertures pour continuer leur projet de manière autonome.

# Présentation d’AGE

## Présentation du produit

AGE est un moteur de jeu vidéo 3D open source, exploitant les dernières fonctionnalités du langage C++ et de l’Application Programming Interface (API) graphique OpenGL. Il est réalisé en partenariat avec AMD, et avec le pôle *Mondes Virtuels* du Hub Innovation d’EPITECH.

Nous souhaitons, à travers AGE offrir aux développeurs un outil de création de jeux vidéo et d'expérience immersive telle que de la réalité virtuelle.

## Les objectifs techniques

Les caractéristiques majeures d'AGE peuvent être décrites selon les points suivants :

* Flexibilité
* Performance
* Innovations graphiques

### Moteur flexible et complet

AGE propose une architecture extrêmement modulable, basée sur le paradigme Entity Component System permettant d'accueillir tout type de jeux vidéo. Effectivement, cette architecture, de plus en plus présente dans les moteurs de jeux modernes, permet au game engine d’accueillir un large éventail de gameplay différents. En effet, son cœur a été pensé pour permettre des utilisations diverses et variées.

De plus, il mettra à la disposition des utilisateurs une série d'outils facilitant le développement de contenus vidéo ludiques de haute qualité (éditeur de scène 3D, éditeur de modèles 3D, ...) et sera compatible avec l'Oculus Rift (casque de réalité augmentée), de manière native et sans développements supplémentaires.

Pour finir, AGE comprendra en son sein une série de fonctionnalités indispensables au développement de jeux vidéo et d'expériences immersives - tels qu'un moteur physique et des effets graphiques avancés.

### Performances

AGE est un moteur de jeu pensé pour apporter de très grandes performances. En effet les technologies qu'il utilise ont été sélectionnées pour leur modernité et leur efficacité. Ainsi, son cœur sera développé en C++11 (puis porté en C++14 lorsque les compilateurs atteindront leur maturité). L'architecture a été pensée pour épouser au mieux les dernières configurations matérielles des ordinateurs (calcul GPU, multithreading, x64).

### Graphismes modernes

AGE se propose d'être un outil innovant graphiquement dans le domaine de l'open source, et plus largement dans l'industrie du jeu vidéo et de la réalité augmentée. En effet, AGE a été pensé pour utiliser les dernières versions de l'API graphique OpenGL (4.3 - 4.4), nous permettant de bénéficier au maximum des derniers pilotes Nvidia et AMD.

De plus, AGE proposera une technique de rendu 3D très rare, le tile forward rendering. Cette méthode, extrêmement récente - et dont beaucoup d'experts s'accordent à qualifier de très prometteuse - n'est pour le moment que très peu présente dans le secteur vidéo ludique.

AGE comprendra en son sein, une méthode de rendu graphique flexible et moderne, en plus de divers effets de qualités, ce qui permettra à l'utilisateur d'ajouter dans son produit une importante quantité de sources de lumière et d'ombres tout en conservant de bonnes performances.

## Les objectifs sur le marché

### Immersion et réalité virtuelle

Persuadés que l'immersion de l'utilisateur occupera une grande partie des enjeux du monde vidéo ludique de demain, nous avons décidé de donner une grande importance à la réalité augmentée, et en premier lieu à l'Oculus Rift.

De plus, nous procéderons à l’intégration du moteur physique open source Bullet, pour permettre aux développeurs de proposer une physique riche et réaliste aux utilisateurs.

### Open Source

AGE est un moteur 3D open source dont le code est disponible à tous pour consultation, édition et utilisation.  
Il est aujourd'hui accessible sur GitHub[[6]](#github) en dépôt public à l'adresse : <https://github.com/Another-Game-Engine/AGE>

L’ouverture du code est rendue possible en grande partie par le choix des technologies utilisées : OpenGL, Bullet, GLM, Cereal, etc. (technologies gratuites et ouvertes).

Nous cherchons actuellement une licence nous permettant de laisser le projet libre tout en nous obtenant une contrepartie financière de la part des utilisateurs souhaitant utiliser AGE à des fins commerciales. Nous nous intéressons de très près à la pratique du multi-licensing qui permet de mettre un projet sous deux ensembles de conditions : l’un pour une utilisation personnelle ou open-source, l’autre pour une utilisation commerciale.

## Les cibles

### Utilisateurs potentiels

Les principaux utilisateurs ciblés par notre moteur sont les développeurs indépendants.

En effet, aujourd'hui la scène française du jeu vidéo n'est pas homogène en termes de techniques utilisées. D’une part, les grands acteurs (Ubisoft, Eugen Software, Quantic Dream) développent des moteurs performants en interne et de manière propriétaire ; d'autre part les plus petites sociétés ont recours à des solutions open source peu chères, ne réunissant jamais flexibilité, performance et rendu graphique. C'est dans un tel contexte qu'AGE prend son sens, en offrant aux entreprises un moteur open source comprenant les dernières fonctionnalités en termes de jeux vidéo 3D.

Ce constat est le même pour la scène de la réalité virtuelle et de l'éducation, le domaine ayant connu une récente explosion venant entre autres de l'évolution des technologies et des matériels (Oculus Rift, Google Glass, Kinect ...).

Ce domaine a besoin de nouveaux outils qui lui permettraient de se développer encore plus vite.

Nous voulons donc développer notre moteur en forte collaboration avec des acteurs présents dans le monde de la réalité augmentée. Nous aurons l'occasion d'approfondir cette étude dans la suite du document.

### Les profils techniques ciblés

AGE a pour objectif d'apporter aux développeurs un moteur de jeu alliant flexibilité de développement, hautes performances et grande qualité graphique.

C'est pourquoi ce projet open source est à destination des développeurs confirmés, maîtrisant le C++ à un niveau avancé.

Néanmoins, il est prévu d’y intégrer une documentation complète ainsi que de nombreux exemples permettant à des développeurs moins expérimentés de tenter leur chance.

### Les plateformes ciblées

AGE est conçu dans un premier temps pour être compatible avec la plateforme Windows. En effet, seule cette dernière permet aujourd’hui l'utilisation des dernières fonctionnalités OpenGL.

Cependant, afin de permettre au plus grand nombre d’utiliser notre moteur dans leur environnement de travail, nous espérons développer la compatibilité Mac et Linux (sous réserve que les derniers pilotes graphiques soient disponibles sur le système d’exploitation).

# Présentation de l’environnement de réalisation

## Environnement de réalisation

### Langages et API utilisés

#### C++11

AGE est développé en C++11, nouvelle norme pour le langage C++. Cette version offre au moteur et à ses utilisateurs une plus grande flexibilité de développement, grâce aux nombreuses nouvelles fonctionnalités, tout facilitant le développement pour les débutants.

Parmi les nouvelles fonctionnalités, celles qui nous intéressent particulièrement sont celles permettant d'augmenter la rapidité d’exécution du code (std::move, std::forward, etc.) ainsi que les outils permettant de se passer au maximum de librairie extérieurs. En effet, dans ce nouveau standard, ont été implémentées des fonctionnalités concernant l'interopérabilité des systèmes tels que les threads. Notre moteur ayant pour but d'être particulièrement performant, ce fut un point important de notre choix vers cette technologie.

#### OpenGL 4.3

Les versions OpenGL 4.x nous permettent de tirer profit des formidables ressources des GPU modernes. En effet un grand nombre de nouvelles fonctionnalités favorisent l'utilisation de la VRAM (mémoire interne au GPU) afin que les données soient plus aisément accessibles par la carte graphique.

De plus certaines de ses fonctionnalités - telles que les "compute shaders" d'OpenGL - permettent de supplanter certaines librairies comme OpenCL[[13]](#opencl) en matière d'utilisation GPU.

Ainsi AGE a pour ambition de suivre l’exceptionnelle croissance du marché des calculs massivement parallèles. Grâce au soutien du constructeur AMD nous avons la possibilité d'avoir des retours sur l'utilisation de ce matériel de plus en plus puissant et prometteur.

#### GLSL

GLSL est un langage permettant l'écriture de shaders[[16]](#shader), nous l'utilisons aujourd'hui dans sa version 400.

Les différences d’architectures entre GPU et CPU nous amènent à réfléchir sur le traitement de nos calculs. Afin de paralléliser au maximum ces derniers, nous utiliserons de préférence les shaders pour pouvoir déporter simplement et efficacement les opérations de traitement.

### Plateformes et outils de développement

#### Windows x64

AGE est développé sur Windows 8.1, dans sa version 64 bits. En effet, les derniers pilotes graphiques sont développés pour la plateforme Windows 8.

De plus, AGE étant pensé pour être utilisé sur une architecture 64 bits (il sera tout de même compatible avec les architectures 32 bits), il est nécessaire de disposer d'un environnement de développement 64 bits.

#### Visual Studio 2013

AGE est développé à l'aide de l'IDE[[8]](#ide) Visual Studio. Nous utilisons la dernière version de celui-ci (2013) car elle apporte de nombreuses fonctionnalités indispensables au projet AGE.

En effet, celle-ci, en plus d'offrir de nouveaux outils de debugging, supporte la compilation du C++11 essentielle au fonctionnement du moteur de jeux.

#### CodeXL

Nous utilisons CodeXL, un outil de debugging sur GPU gratuit, développé et maintenu par l'entreprise AMD.

Ce logiciel offre beaucoup de fonctionnalités essentielles au développement sur carte graphique, parmi lesquels :

* Détections des erreurs OpenGL
* Break points
* Edit and continue sur les shaders
* Inspection des FBO[[4]](#fbo) et des VBO[[19]](#vbo) au runtime

#### Git

Git est un outil de gestion de versions, spécifiquement conçu pour faciliter d'une part la sauvegarde de l'historique de production et d'autre part, la collaboration entre développeurs sur un même projet.

## Environnement matériel

### Environnement de développement

#### Ordinateur moderne

AGE étant destiné aux architectures matérielles modernes, il est nécessaire pour son développement de travailler sur un ordinateur récent.

Par conséquent, la majorité des développeurs d'AGE disposent d'un ordinateur réunissant :

* Un processeur multicœur (Intel I5 / I7) accélérant le temps de compilation.
* Un disque dur SSD apportant à Visual Studio 2013 un meilleur temps de réponse pour l'auto complétion.
* Une carte graphique récente et puissante permettant le calcul GPU et l'affichage d'effets complexes à l'écran.

#### Oculus Rift

L'Oculus Rift étant intégré au projet, il nous est nécessaire de disposer d'un Oculus pour le développement d'AGE. Par conséquent, nous avons commandé un kit de développement Oculus, et sommes en attente de sa livraison.

L'Oculus que nous allons utiliser sera le dernier prototype (Mars 2014). Cela nous permettra d'intégrer à AGE les dernières fonctionnalités offertes par le produit.

#### Périphérique de contrôle

De manière à augmenter l'immersion des joueurs, AGE intégrera le support de la manette Xbox. Par conséquent l'équipe de développement du moteur a en sa possession plusieurs de ces produits.

## Architecture technique

### Le moteur de jeu

#### L’architecture Entity Component System

AGE repose sur une architecture dite “Entity Component System Based”. Cette dernière de plus en plus utilisée dans le développement de jeux vidéo permet de réaliser un grand nombre de types de jeux différents, sans pour autant réduire les performances de ces derniers.

Nous tenterons d’expliquer dans les points suivants la forme générale de l’architecture.

##### Les entités

Les entités sont les objets premiers du jeu vidéo, en d’autres termes, elles représentent chaque élément distinct dans un jeu (un texte, un personnage, une lumière, une table, un vase…).

Elles doivent être comprises comme des conteneurs, vides dans un premier temps, ne disposant que d’une information de transformation spatiale et disposés dans un arbre hiérarchique.

##### Le graphnode, ou arbre hiérarchique

L’arbre hiérarchique, plus communément appelé “graphnode” est composé d’entités (les nœuds), ses branches représentant les liens entre les entités. Ainsi, une entité A pourra avoir deux entités filles (B et C) et une entité parente (Z).

Cette organisation des entités dans un arbre permet d'opérer des opérations de transformation groupées, par exemple, une entité voiture, ayant comme enfant une entité conductrice, déplacera cette dernière lorsqu'elle sera amenée à bouger.

##### Organisation en mémoire

Le processeur de l’ordinateur est responsable du traitement des données, en effet, il applique les calculs que le programme lui demande d’exécuter sur une série de données.

Il est connu que le processeur gagne en rapidité de calcul lorsqu’il effectue ses opérations sur des données concomitantes en mémoire (les unes à la suite des autres dans la RAM). C’est pourquoi les entités seront regroupées au sein du même espace de la mémoire de l’ordinateur, ainsi, le processeur réalisera les calculs nécessaires sans effectuer de sauts dans la RAM, ce qui permettra à AGE d’offrir à ses utilisateurs de plus grandes performances.

##### Les composants

Les entités comme nous l’avons expliqué plus haut se limitent dans leur définition à être des conteneurs vides dans un premier temps, ne disposant que d’information de transformation et d’organisation hiérarchique.

Les composants (ou “components” en anglais) vont jouer un rôle très important dans la spécialisation des entités.

Les composants sont de simple structure de données (“plain old data structure”). Chaque type de composant embarque en son sein une série de données spécifiée par le développeur. Ainsi, il existera des composants de type “Objet de rendu 3D” qui contiendront toutes les informations nécessaires au rendu d’objet 3D ; d’autre de type “Point lumineux” décrirons les qualités d’une lumière (“point light”), parmi lesquelles son rayon, sa couleur et son intensité.

Les composants sont associés aux entités. En effets, en étant contenus dans une entité, ils participeront à la spécialisation de cette dernière. En d’autres termes, les composants définiront l’essence des entités. Ainsi, nous pouvons imaginer une entité ayant des composants “Objet de rendu 3D” et “Point lumineux” ; elle sera alors représentée au sein du jeu comme un objet 3D émettant de la lumière.

##### Les systèmes

Les entités jouent donc le rôle de conteneurs et les composants définissent leur essence. Les systèmes quant à eux vont dicter aux entités leur comportement, en fonction des composants qu’elles contiennent.

Les systèmes sont spécialisés dans un type d’entité. Ainsi, ils ne traiteront que les entités contenant la suite de composant qu’ils requièrent.

Les systèmes sont informés en temps réel par le moteur de chaque modification apportée sur les entités (ajout ou suppression d’un composant). Lorsqu’une entité correspond à leur prérequis, ils l’ajoutent à leur liste des éléments à traiter.

Lorsque la fonction de mise à jour des systèmes est appelée, chaque système effectue une suite de calcul sur chacune des entités présentes dans sa liste.

#### Chargement et sauvegarde

La sauvegarde et le chargement des données constituent une exigence fondamentale du jeu vidéo.

En effet, il doit être possible à tout moment de sauvegarder sur le disque dur l’état d’une partie, pour pouvoir le charger plus tard, sans que le joueur ne remarque de modification.

Ainsi, AGE propose un système de chargement et de sauvegarde intègre. Ce dernier permettant de sauvegarder la composition de chaque entité et l’état de chacun de ses composants.

##### Sauvegarde multiformat

La sauvegarde des données du jeu sera proposée sous plusieurs formats.

Premièrement, en binaire, ce qui offrira aux développeurs de meilleures performances lors de l’écriture et de la lecture, ainsi qu’une meilleure compression de données.

Cependant, les données étant écrites en binaire, il est impossible de les éditer à la main. C’est pourquoi AGE offrira aussi la possibilité de sauvegarder et de charger des données sous le format JSON. En effet, ce dernier est textuel et son édition est possible à travers n’importe quels éditeurs de texte.

#### Gestion réaliste de la physique

Pour permettre un plus large éventail de type de jeux possibles, et pour offrir un plus grand réalisme et par conséquent une meilleure immersion du joueur, AGE proposera une intégration de la physique réaliste.

##### Intégration des rigid bodies

AGE proposera d’associer à des entités des propriétés physiques, plus particulièrement, des formes de collisions.

Ces dernières, plus communément appelées “rigid body”, décrivent une forme à partir de laquelle pourront être calculées les collisions. Elles peuvent prendre différents aspects : sphère, cube, mais aussi des formes beaucoup plus complexes, offrant ainsi une plus grande précision et un meilleur réalisme.

##### Informations de collision

Associées aux formes de collision, des informations d’impacts seront mises à la disposition du développeur. Il pourra utiliser ces dernières pour implémenter différentes mécaniques de gameplay. Par exemple, si une entité “Projectile” entre en collision avec une seconde de type “Personnage”, le développeur pourra en fonction de la force de l’impact modifier le niveau de vie de ce dernier.

##### Character controller

AGE proposera une intégration complète du “character controller” au sein de son moteur physique. Le “character controller” permet de prendre le contrôle sur une entité du jeu et de voir cette dernière réagir de manière crédible dans ses déplacements. Cette fonctionnalité est présente dans la totalité des jeux de type FPS (First Person Shooter) où l’on peut se déplacer dans un monde. Ainsi le personnage ne peut sauter au-dessus d’obstacle dépassant une certaine hauteur et ne peut traverser les murs par exemple.

Cette implémentation permettra à AGE d’offrir un meilleur réalisme. En effet, comme nous le verrons plus tard, la vue subjective est au cœur du fonctionnement de l’Oculus Rift et de plus généralement de la réalité virtuelle. Par conséquent, pouvoir offrir la possibilité de se déplacer dans un monde, d’un point de vue subjectif et de manière physiquement réaliste augmentera le degré d’immersion proposé par AGE.

#### La gestion des assets

Le terme “assets” regroupe tous les fichiers nécessaires à la production d’un jeu vidéo 3D, parmi lesquels, les modèles de rendu, mais aussi les textures, les animations ou les polices.

AGE proposera à l’utilisateur un large éventail d’assets de types différents, permettant ainsi aux développeurs de réaliser un jeu riche et complet. Parmi ces assets, AGE proposera une intégration native de ces derniers :

* Modèle de rendu
* Textures
* Animations
* Matériaux (structure de données décrivant les propriétés de rendu des objets)
* Les sprites[[18]](#sprite) (permettant d’intégrer des animations 2D)
* Les polices
* Les modèles de collisions (forme de collision précalculée)

Nous verrons plus tard que ces différents assets seront convertis dans un format spécifique à l’aide d’un outil de notre conception. La conversion, ou sérialisation, dans format dont nous aurons défini les propriétés, permettra un chargement beaucoup plus rapide de ces dernières.

#### Le son

AGE proposera une intégration complète du son permettant de jouer des effets sonores ainsi que des pistes audio.

##### Son 3D

AGE intègrera une gestion 3D du son, de manière à augmenter l’expérience immersive du joueur.

##### Sound emitter / Sound receiver

Un composant de type *sound emitter* sera proposé. Il permettra d’attacher un émetteur de son à des entités spécialisées, à la manière d’un haut-parleur

D’un autre côté, une composante de type *sound receiver* permettra d’ajouter aux entités la possibilité de recevoir le son environnant.

#### Utilitaires

Une série de fonctions utilitaires seront mises à la disposition des développeurs pour leur faciliter de développement. Ainsi, des algorithmes de génération aléatoire et de “range” seront intègres à AGE.

De plus, une dépendance permettra d’enregistrer des configurations pour le produit réalise à l’aide d’AGE. Ces configurations seront éditables lors de la partie et automatiquement enregistrées dans un fichier sur le disque dur. En d’autres termes, il sera possible très simplement pour les développeurs de mémoriser des informations indispensables à son application telle que la résolution de l’écran, le volume du son ou les préférences d’inputs.

#### Benchmarks

Les performances étant au cœur de la réalisation de jeu vidéo et d’expériences 3D interactives, nous mettrons en place un système d’évaluation de ces dernières.

Ainsi, le développeur pourra lors de l’exécution évaluer la consommation de ressource de chacune des parties de ce dernier. Parmi lesquelles :

* Le nombre de frames[[5]](#frame) par seconde
* Le temps pris par rendu
* Le nombre d’entités active
* Le nombre d’entités visible
* Le temps consommé par le processeur par tour de boucle
* Le nombre de formes de collision
* Le temps consommé par chaque système
* La quantité de mémoire vidéo allouée

### Graphical engine

#### Frustrum Culling

In order to increase engine performance, it is vital to make a culling non-visible surfaces by the camera.

Thus, regardless of the number of polygons in our scene frame rate will remain fluid because we calculate only the display of objects visible by the camera.

#### High Dynamics Range (HDR)

The HDR is an effect to dynamically simulate the operation of the retina at a glare. This effect is conducive to the immersion of the player in dark areas and bright thanks to its highly realistic visual impact.

#### Glow

The "Glow" or "Blur bright" is a post-effect (that is to say, practiced on the image to final render) called "neon" and is to blur the area surrounding the subject object to this property.



Figure 1 : Example of Glow

#### Dynamic Shadow Mapping

Technique for establishing realistic shadows dynamically calculated on all types of light.

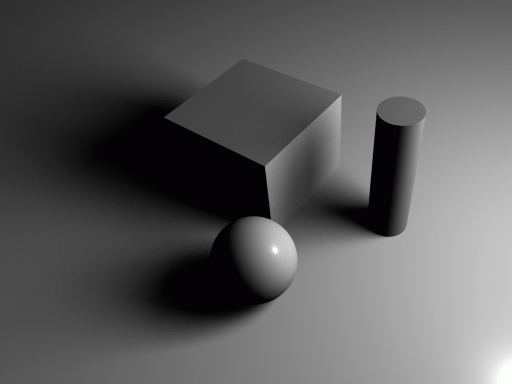


Figure 2 : Dynamic Shadow Mapping

#### Reflexion Mapping

Technique using a Cube Map to simulate the visual mirror and this on any type of object.



Figure 3 : Reflexion Mapping on a sphere

#### Anti-Aliasing

To reduce aliasing in our records, we will apply the algorithm to drastically reduce FXAA pixelation edges of our geometries.



Figure 4 : Same object on the left without FXAA and with FXAA on the right

#### Tile Based rendering

The "light tile based culling" is a method of culling of lights which calculates the influence thereof on only the areas of the screen (Tile) where they are needed.

This allows to significantly reduce the computational complexity of illumination of a scene, regardless of the number of dynamic lights present.

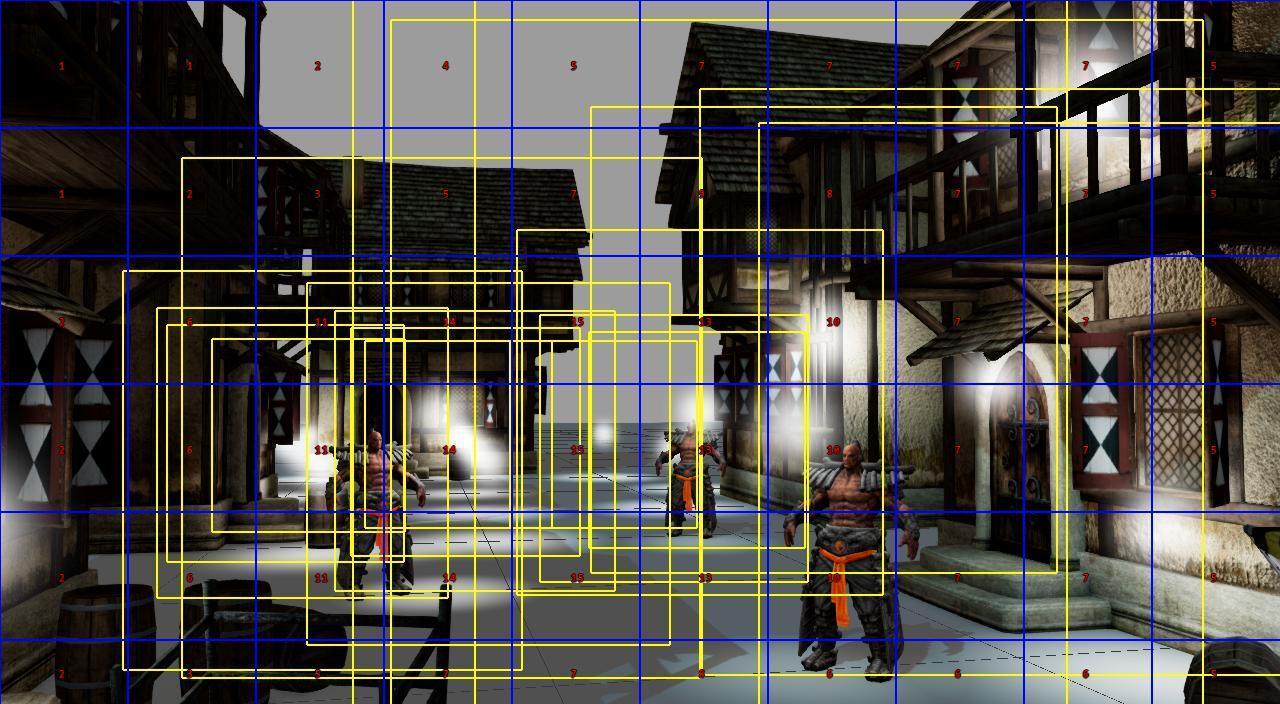


Figure 5 : Effet des lumières sur l'image en fonction des tiles

##### Deffered rendering

We will couple the technique of "tile based rendering" with a rendering method called "Deffered rendering" for calculating lights only on the pixels actually displayed.

Thus, if a pixel passes the test light culling, but not part of the final report because it is hidden by such a geometry that is displayed above the influences of different lights will not be calculated for this above.

This effect also allows the implementation of many algorithms, including some global illumination algorithms (simulating indirect illumination in a scene).

##### Foward rendering

We also propose a rendering method "classic", especially to make semi-transparent objects in our scenes.

This method will just make all objects in the scene, regardless of whether it is hidden by any geometry.

#### Normal mapping

Normal mapping is a graphical effect to simulate reliefs on surfaces in a 3D environment, and without having to increase the number of polygons in the scene.

The basic principle is to apply a texture to a 3D object having charge of changing the way it will be illuminated.

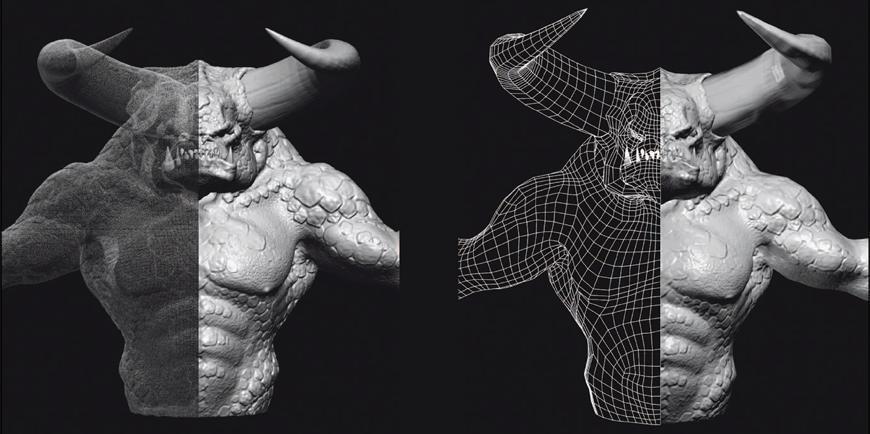


Figure 6: on the left: image obtained with high polygones mesh, on the right obtained with normal mapping

(same result but faster to draw).

### Publishing tools

#### Scene Editor

A level editor is available to developers. The latter meet several features that facilitate the production of a video game or a 3D scene.

The publisher would propose initially to place the entity in space, as well as configure the various components they contain.

In addition to publishing scene, this editor will offer the possibility to create and save "prefabs". These are nothing but a collection of entities preconfigured, which facilitates subsequent development. Thus, we can imagine a prefab "enemy" during the course of the game, simply call the creation of this type of prefab an enemy to appear in the scene.

#### Assets convertor

A converter assets will be proposed in the context of the use of the AGE engine. The latter propose to convert native files, files compatible with the motor.

During this conversion data will be optimized to allow faster loading and provide better engine performance.

The assets converter will take different input file types including .obj, .fbx, .dae, .tga, .jpg, .png, .mtl, .bullet, .ttf. From these, it will generate serialized files.

Over the assets converter will generate forms of collision objects from 3D rendering.

## Existing Components

### Technologies integrated in the motor

#### SDL

SDL is a free and open-platform graphics library. It is used in EGM under the window creation and management of keyboard / mouse events.

#### Bullet

Bullet is a physics engine for managing interactions between different body in a 3D environment. It includes among its many features:

* Collisions
* Rigid body
* Soft body
* Culling d'objet

Just like SDL, it is under a free license zlib.

#### FBX SDK

FBX SDK is a development kit that allows loading FBX model. The latter is a file format containing all the information necessary for the realization of 3D object. It includes :

* color
* normal
* texture
* animation

Developed by Autodesk it is now one of the most used in the world by designers and 3D animation formats. This SDK also allows us to load 3D models collada and obj type, or the most popular formats among graphic designers.

#### Cereal

Cereal is an open source project providing access to a number of practical tools, serialization and deserialization of data structures. We used it as part of the saving and loading information.

#### Oculus SDK

The oculus rift is a helmet augmented reality for the general public. This tool presents itself as the central device to our goal of immersive. Aiming to enhance user experience in virtual worlds, the oculus rift is an opportunity for AGE enhance its graphics and physical effects of an innovative and immersive way..

#### FMOD

FMOD is a proprietary library useful in the management of 3D sound. It is one of the most used tools in the world of real-time 3D.

#### GLM

GLM is a math library designed for use with OpenGL. It allows us to perform all our calculations of transformation.

#### FreeType

FreeType is an open source library for easy loading fonts. We use it as part of the asset converter to generate the images that will be used for text.

### Technologies of Project Management

#### Github

##### Issues – Milestones

To facilitate communication during development, we will use the system after and Milestones proposed by GitHub. Each new implementation, updating or debugging will be preceded by an issue assigned to the team member concerned. This will allow a good view of all the tasks to be slaughtered and their realization.

The Milestones allow you to place these issues in the context of time to impose deadlines on various parts of the project.

##### Pull request

Each major new feature will be validated by the group before being mergée at the heart of the project, in order to guarantee the stability of the final code.

To do this we will use the system pull request and branch offered by GitHub.

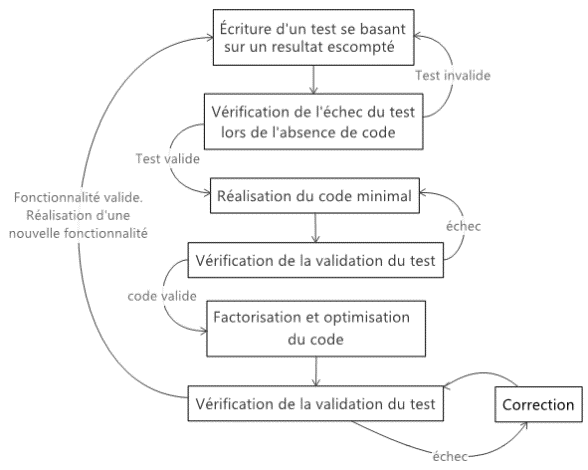
La base du projet sera sur la branche dite "master". Le reste du développement se fera sur des branches parallèles qui n’influeront pas sur la branche principale avant la réunion grâce à la pull request.

The basis of the project will be on the so-called "master" branch. The rest of the development will be on parallel branches which do not affect the main branch before the meeting with the pull request.

## Tests stratégie

The development of the engine is accompanied by a test strategy, consisting of four types of tests to both retain a permanent code, and to meet the requirements of our future users.

### Units tests

Unit testing ensures the stability of the project, they will be made throughout the development and will be developed for each function and member function.

We use the Google Test Framework, an easy to use open-source tool provided by Google. We will use in applying TDD (Test Driven Development) approach that advocates writing unit tests before writing the source code of the function. This allows first time to ensure that the code in question is unitarily testable. TDD cycle present in all four to six steps:

Figure 7: Schéma du cycle TDD

### Functional tests

These tests ensure compliance capabilities vis-à-vis the need to which they respond. To do this, we will design and create some games using the features to be tested to ensure we Usability AGE. They will be made when the engine is sufficiently advanced to offer maximum functionality to be tested.

### Acceptation tests

Afin d’éprouver les capacités de notre moteur, il sera utilisé par des développeurs durant des événements tels que des hacakatons. Ces types de test permettent d’avoir un véritable retour utilisateur sur notre produit afin de pouvoir corriger ses éventuelles imperfections techniques ou ergonomiques.

### Performance tests

To test the performance of our engine, we use certain programs to evaluate the use of equipment. CPU level, we use the benchmark tool Visual Studio 2013. For the GPU, we use "NSight" (Nvidia software to evaluate the performance of OpenCL and OpenGL contexts) and XL Code (software DMA) to have visibility into resources (Vertex Buffer, Frame Buffer, Texture).

## Communication

The project is an open-source engine, so it is important for us to develop the communication aspect at the beginning of the realization, on the one hand to be able to collect all the different user feedback throughout the development, and secondly to enable and support the event.

### Visual identity

The visual identity is a pillar of a communication strategy. It transmits an image directly easily and quickly identifiable. It enables the transmission of values ​​and a mindset for people who can come together and join our identity. It is therefore important for us to create this identity in the form of a logo and a graphic for the website.

### Social networks

#### Facebook

A page dedicated to our project will allow us to create and unite a future community of testers and create a quick link with partners. The Facebook page will be used primarily as a tool for social communication, with a weekly rate of publication on our work, but also exchange page on which key to our project.

#### Twitter

It is a parallel solution to reach a professional audience and to communicate announcements demos, contests, or events within Epitech (eg Hackathon). The platform will be a way for us to communicate in a timely manner. For example, creating a Hashtag for an event and promote it, or by initiating and encouraging discussion around AGE to create buzz.

#### LinkedIn

Using LinkedIn helps maintain links between professionals, it can be considered as a knowledge network. Indeed, the advantages enjoyed by the people on the network are the exchange of information, ideas and opportunities. These points are essential for us! Invest time in LinkedIn is important since it is a network that is designed with the business.

### Le site

The heart of our communication will have course our website. The website development has many points to meet up with all the communication issues that our project will face.

#### Technologies

We will use Ruby, Javascript and HTML technologies to create our site. These technologies are very focused on productivity and simplicity, this will allow us to reduce the development time of the site to really begin as soon as possible in our communication project.

#### Architecture

The site will offer:

A home page for important information about our project, news, important links.

A section "Downloads" to manage different resources to be brought to be shared with the engine operator. Of course this topic is brought to live with the GitHub repository which we work and is also a platform resources. This section will be reserved for so-called "fixed" or requiring more bandwidth for assets such resources.

A forum. The latter is necessary for two reasons: the first allows the management support for our engine. It is essential that users can come and ask questions, share experiences, and find help when they need it. Next, technical communication will be the main function of the forum.

A "Support" section, this section is necessary for any documentation that we provide users of the project. The documentation will have to evolve during the project, a discount work will be required. Documentation for more focused "research and development" purposes, describe the operation of some more complex implementations such as "Tile Based Rendering". In order to submit to an interested feedback on problems encountered, solutions provided and results obtained. This section is one of the most important site in terms of workload.

### Wiki

Creating a Wiki can be a knowledge base and dynamic resource. It is a collaborative public workspace that feeds free public input. It will contain a detailed engine architecture plan and its various features and their use. This documentation is intended to help potential users to take control of the game engine

## Project Organisation

### An agile method : Scrum

The team is organized using methods Say "agile", specifically using Scrum.

The latter is perfectly suited for small teams, which is why we have chosen for the development of AGE. It is based on dividing the project in increments called sprints. the project grows by incorporating features expressed through the backlog using an iterative process.

Scrum requires strong communication, so the team will meet at a scrum several times a week for everyone expresses the progress of his part of the work by answering these three questions:

* What have I done this week?
* What should I do the next week?
* What are the difficulties?

There are two key roles more to allow the establishment of Scrum:

The *product owner*, or owner of the product, means the member of the team that has the vision of the project. It takes major decisions involving the direction of the project and defines the order in which the features are developed. Finally, from a technical point of view, it ensures that the backlog is understood by the entire team, for at any moment, everyone knows what to work perfectly.

The *scrum master*, refers to her team member who is responsible for the method. It is responsible to optimize the capacity of the development team, helping to apply Scrum to work independently, and to overcome obstacles and unexpected slowing work. He also teaches the Product Owner writing different parts of the backlog.

The rest of the development team does not have predefined roles, it is self-organized and multidisciplinary. There are no internal hierarchy and all decisions are taken together.

### Internal team organisation

The team currently consists of seven people, was organized in this way:

laureline Patoz

* Scrum Master.
* Development on the engine of the game.
* Development on the part demos, gameplay.

César Leblic

* Product Owner.
* Referent and developer on the game engine part.
* Development Tools Part.

Anthony Boucher

* Development on the game engine.

Paul Baron

* Referent and developer on the game graphics rendering.

Dorian Pinaud

* Referent and developer of the OpenGL pipeline.
* Development of the game graphics rendering.

Philippe-Henry Sutter

* Communication (events, website, tutorials, etc.).
* Development of the motor part of game.
* Development of the demos part gameplay.

Théo Bolognini

* Communication (events, website, tutorials, etc.)
* Development of the motor part of game.
* Development of the demos part gameplay.

# Annexes

## Glossary

1. *Backlog* - The backlog, product backlog or in French, is a list of features or tasks presented atomically to meet the needs of users. This is a key element of Scrum. [**↨**](#a_backlog)
2. *Cube map*  - A cube-shaped texture, that is, six faces of size oriented in space. [**↨**](#a_cubemap)
3. *Culling* - Culling is a technique to display only the visible polygons. And hidden faces are not calculated.[**↨**](#a_culling)
4. *FBO* - An FBO or Frame Buffer Object OpenGL is a tool to keep a frame rendering to be able to ask some treatments before the final display.[**↨**](#a_fbo)
5. *Frame* - This is an image displayed on the screen during a given time corresponding to the rendering of a graphics engine. The frame rate corresponds to the frame rate per second. The actual video games generate up to 200 frames per second.[**↨**](#a_frames)
6. *GitHub* - It is a hosting service and management of software development based online management software decentralized Git version.[**↨**](#a_github)
7. *Hashtag* - This is a marker of metadata used on the Internet that allows you to associate a keyword content to share. It begins with a spider "#" hash English.[**↨**](#a_hashtag)
8. *IDE* - An Integrated Development Environment (IDE) is a development environment containing all the essential tools for programming: text editor, benchmarking tools, compiler and debugger. [**↨**](#a_ide)
9. *Issue* - Similar to a ticket, the outcome is used to represent a problem that must be solved. This can be both a feature to implement a bug to remove.[**↨**](#a_issue)
10. *Merge* - The data merge unit according to one of the similarity criteria.[**↨**](#a_merge)
11. *Méthodes agiles* - These are groups of practices project emerged in 1991 can be applied to various types of projects. They want to be more pragmatic than traditional development methods involving up the applicant and authorizing unexpected during the development phase.[**↨**](#a_agile)
12. *Milestone*  - Ou jalon en français, est la fin d’une étape, ou représente un évènement important dans le cadre de la gestion de projet.[**↨**](#a_milestone)
13. *openCL* - is an API designed for programming heterogeneous parallel systems such as GPU and multi-CPU heart. [**↨**](#a_opencl)
14. *Pull Request*  - A pull request or demand draft, Inclusion of a branch in the main branch of a deposit. [**↨**](#a_pullrequest)
15. *Un scrum* - In Scrum, a scrum is a regular and very short meeting of all members of the team and making the inventory of the latest works. [**↨**](#a_scrum)
16. *Shader* - A shader is a computer program that sets some of the rendering process. Thus, effects or surfaces that seem complex can be made from simple geometries. [**↨**](#a_shaders)
17. *Sprint* - A sprint is a term used in the Scrum agile method and represents the increments that make up the project. They can last from several hours to one month (with a preference for two weeks). Each sprint begins with an estimate subsequent operational planning. The sprint ends with a demonstration of what has been completed, and helps to increase the business value of the product. Before starting a new sprint, the team conducts a retrospective: it analyzes what happened during the sprint to improve for the next. [**↨**](#a_sprint)
18. *Sprite*  - A denotes a sprite image in two dimensions which can be moved, or can undergo further processing, with respect to the screen. [**↨**](#a_sprites)
19. *VBO*  - English Vertex Buffer Object is an OpenGL method to send data to the 3D graphics card. VBOs allow better performance because the data is no longer in the system memory, but resident in the GPU. With this technique, the rendering is faster. [**↨**](#a_vbo)