~AGE~



Technical specifications v3.0

15/06/2014

César Leblic

Laureline Patoz

Anthony Boucher

Dorian Pinaud

Paul Baron

Philippe-Henry Sutter

Théo Bolognini

# 

# Abstract

This document presents the third version of the technical specifications for our Epitech Innovative Project named AGE (Another Game Engine). It will be used as a guideline for the team in charge of this project and lists all the needs, the requirements and the constraints that we will need to follow during the implementation of the project.

In the first place, we will depict the whole project, and then, step by step, we will describe the targets aimed by this game engine and the tools that will be used during its development. The description of the different parts of the software will picture more precisely the work that will be done. Finally, the tools which will allow the team to have an optimal organization are listed and the communication aspect that is essential to the project is developed.

# Document description

|  |  |
| --- | --- |
| Title | CDC AGE |
| Date | 15 juin 2014 |
| Authors | César Leblic, Laureline Patoz, Anthony Boucher, Paul Baron, Dorian Pinaud, Philippe-Henry Sutter, Théo Bolognini |
| E-Mail | [ageproject@googlegroups.com](mailto:ageproject@googlegroups.com) |
| Subject | Technical specifications AGE |
| Version | 2.0 |

# Revisions table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Date | Version | Author | Section(s) | Comments |
| 22/04/14 | 1.0 | Paul Baron | / | Version 1.0 of the document |
| 01/06/14 | 1.1 | Anthony Boucher | / | Corrections |
| 01/06/14 | 2.0 | Laureline Patoz | / | Document layout |

Summary

[I. Epitech presentation 1](#_Toc386195042)

[A. The school 1](#_Toc386195043)

[B. Presentation of the EIP 1](#_Toc386195044)

[II. Presentation of AGE 1](#_Toc386195045)

[A. Product presentation 1](#_Toc386195046)

[B. Technical goals 2](#_Toc386195047)

[1. A flexible and complete engine 2](#_Toc386195048)

[2. Performances 2](#_Toc386195049)

[3. Modern graphics 2](#_Toc386195050)

[C. The objectives on the market 3](#_Toc386195051)

[1. Imersion and virtual reality 3](#_Toc386195052)

[2. Open Source 3](#_Toc386195053)

[D. The targets 3](#_Toc386195054)

[1. Potential users 3](#_Toc386195055)

[2. Targeted technical profiles 4](#_Toc386195056)

[3. Targeted platforms 4](#_Toc386195057)

[III. Presentation of the working environment 4](#_Toc386195058)

[A. Working environment 4](#_Toc386195059)

[1. Used languages and APIs 4](#_Toc386195060)

[2. Platforms and devlopement tools 5](#_Toc386195061)

[B. Hardware environment 6](#_Toc386195062)

[1. Developement environment 6](#_Toc386195063)

[C. Technical architecture 6](#_Toc386195064)

[1. The game engine 6](#_Toc386195065)

[2. The graphical engine 10](#_Toc386195066)

[3. The edition tools 14](#_Toc386195067)

[D. Existing components 14](#_Toc386195068)

[1. Technologies integrated to the engine 14](#_Toc386195069)

[2. Project managment Technologies 16](#_Toc386195070)

[E. Test stategy 16](#_Toc386195071)

[1. Unit tests 16](#_Toc386195072)

[2. Fonctionnal tests 17](#_Toc386195073)

[3. Acceptation tests 17](#_Toc386195074)

[4. Performance tests 17](#_Toc386195075)

[F. Communication 17](#_Toc386195076)

[1. Visual identity 17](#_Toc386195077)

[2. Social networks 17](#_Toc386195078)

[3. The website 18](#_Toc386195079)

[4. Wiki 18](#_Toc386195080)

[G. Project management 19](#_Toc386195081)

[1. Agile methodology: Scrum 19](#_Toc386195082)

[2. Internal team organisation 19](#_Toc386195083)

[IV. Annexes 21](#_Toc386195084)

[A. Glossary 21](#_Toc386195085)

# Presentation of EPITECH

Epitech is a school which train expert in IT and new technologies. This school, a member of IONIS group, is spread over 12 sites in France. Its goal is to train computer enthusiasts with an innovative pedagogy focused on real business needs and the learning modes of the new generation.

In addition to the mandatory projects throughout the schooling, students create a personal project of graduation: the Epitech Innovative Project (EIP). The choice of subject is free, as long as the basic concept is innovative and technically interesting. Students then form a group of at least six students of their promotion to realize this project from the third to the fifth year. The success of the EIP will be assessed on their ability to design, produce and present their project. Throughout its implementation, the labEIP offers many interviews to follow and to guide groups in their progression, and enable them to lead a project in its entirety, with among others, the establishment of a technical specification document, the integration of an analysis of what already exists in this field, the conception, the implementation and the communication around the project, the marketing aspect, etc. Thus, at the end, the project must be functional and able to be used in the professional world.

The purpose of this innovative work is to provide to Epitech’s students practical project management and experience of teamwork. This gives students a solid foundation for their entry into the professional world, but also an opportunity to continue their project independently.

# Presentation of AGE

## Presentation of the product

AGE is an open source 3D game engine which use the last functionalities of the C++ langage and of the graphical Application Programming Interface (API) OpenGL. This project is developed in partnership with AMD and the Virtual reality pole of the Epitech’s innovation Hub.

We hope, through AGE, offer to developers a tool for creating video games and immersive experiences using virtual reality.

## The technical goals

The main characteristics of AGE can be described according to the following points:

* Flexibility
* Performance
* Graphic innovation

### A flexible and complete engine

AGE provides a highly scalable architecture based on the Entity Component System paradigm to accommodate any type of video games. This architecture, which is more and more present in modern engines, allows the game engine to accommodate a wide range of different gameplay. Indeed, its core was designed to allow various uses.

In addition, it will provide for users a set of tools facilitating the development of high quality playable content (3D scene editor, editor of 3D meshes ...) and will be compatible with the Oculus Rift (headset for augmented Reality) natively without additional development needed.

Finally, AGE will include within it a series of features essential to the development of video games and immersive experiences - such as physics engine and advanced graphics.

### Performances

AGE is a game engine designed to provide very high performance. In fact, the technologies it uses have been selected for their modernity and efficiency. Thus, its core will be developed in C++11 (then brought to C++14 compilers when they will reach the necessary maturity). The architecture has been designed to best fit the latest hardware configurations of computers (GPU computing, multithreading, x64).

### Modern graphics

AGE aims to be an innovative tool graphically in the field of open source, and more broadly in the gaming and augmented reality industry. Indeed, AGE has been designed to use the latest versions of the graphic API OpenGL (4.3 - 4.4), allowing us to take maximum advantage of the latest Nvidia and AMD drivers.

Furthermore, AGE propose very rare 3D rendering technic, the tile based forward rendering. This extremely recent method - and many experts agree to describe it as very promising - is currently very little used in the video game industry.

AGE will include within it a flexible and modern graphic rendering method, with a variety of quality effects, allowing the user to add to his product a significant amount of light sources and shadows while maintaining good performances.

## Objectives on the market

### Immersion and virtual reality

Convinced that the immersion of the user will occupy a large part of the challenges of tomorrow's video game industry, we have decided to give great importance to augmented reality with the integration of hardware such as the Oculus Rift.

In addition, we will proceed with the integration of the open source Bullet physics engine, allowing developers to create rich and realistic physical worlds.

### Open Source

AGE is a 3D engine open source engine which code is available to all for viewing, editing and use. It is now available on github as a public repository at the following address: <https://github.com/Another-Game-Engine/AGE>

The opening of the code is made possible largely by the choice of technologies: OpenGL, Bullet, GLM, Cereal, etc.. (free and open technologies).

We are currently looking for a license to let the project open source while getting a financial contribution from users wishing to use AGE for commercial purposes. We look closely at the practice of multi-licensing that puts a project under two sets of conditions: one for personal use or open-source, the other for commercial use.

## The targets

### Potential users

The main users targeted by our engine are independent developers.

Indeed, today the French video game scene is not homogeneous in terms of techniques used. On the one hand, the major players (Ubisoft, Eugen Software, Quantic Dream) develop efficient proprietary game engines internally; other smaller companies use cheap open source solutions, never bringing flexibility, performance and graphics rendering. It is in this context that AGE makes sense, giving businesses an open source engine including the latest features in terms of 3D video games.

This observation is the same for the scene of virtual reality and education, the field has seen a recent explosion coming from the development of technologies and hardware (Oculus Rift Google Glass, Kinect ...).

This area needs new tools that would allow it to grow even faster.

We want to develop our engine in strong collaboration with actors of the augmented reality field. We will have the opportunity to explain that later in this document.

### Targeted technical profiles

AGE aims to provide developers with a game engine that combines development flexibility, high performance and high quality graphics.

That is why this open source project is aimed at experienced developers who are familiar with C++ at an advanced level.

Nevertheless, it is planned to integrate a comprehensive documentation and numerous examples to less experienced developers to try their luck.

### Targeted platforms

AGE is designed initially to be compatible with the Windows platform. Indeed, windows is currently the only platform to support OpenGL 4.

However, to allow the greatest number to use our engine in their work environment, we hope to develop the Mac and Linux compatibility (provided that the latest graphics drivers are available on this operating systems).

# Presentation of the working environment

## Working environment

### Used API and languages

#### C++11

AGE is developed in C++11, the new standard for the C++ language. This version offers to the engine and its users greater flexibility of development, thanks to numerous new features, while facilitating the development for beginners.

Among the new features, those that particularly concern us are those that increase the speed of the execution of code (std::move, std::forward, etc..) As well as the tools to avoid as much as possible the use of external libraries. Indeed, in this new standard features provided by the C++11 specification have been implemented some concerning the interoperability of systems such as threads. Our engine designed to be particularly effective, this was an important point for our choice of this technology.

#### OpenGL 4.3

OpenGL 4.x versions allow us to benefit from the tremendous resources of modern GPUs. Indeed, a large number of new features enhance the use of VRAM (GPU internal memory) so that the datas are more easily accessible by the graphics card.

Furthermore some of its features - such as OpenGL "compute shaders"- possible to override some libraries like OpenCL in GPU computing.

And AGE aims to follow the exceptional growth of the market of massively parallel calculations. With support from the manufacturer AMD we have the opportunity to get feedback on the use of this material increasingly powerful and promising.

#### GLSL

GLSL is a language for writing shaders. We are using today the version 400 of this language.

Differences between GPU and CPU architectures lead us to reflect on the treatment of calculations. To parallelize up these, we preferably use shaders to be able to simply and effectively deport processing operations.

### Platform and development tools

#### Windows x64

AGE is developed on the 64-bit version of Windows 8.1. Indeed, the latest graphic drivers are developed for the Windows 8 platform.

Furthermore, AGE is thought to be used on 64-bit architecture (it will still be compatible with 32-bit architectures), it is necessary to have a development environment for 64-bit.

#### Visual Studio 2013

AGE is developed using the Visual Studio IDE. We use the latest version of this (2013) because it provides many essential functionalities that we are using during the development of the AGE project.

Indeed, in addition of providing new tools for debugging, it supports compilation of the C++11 specification, essential to the AGE game engine.

#### CodeXL

We are using CodeXL, a free GPU debugging tool developed and maintained by AMD.

This software offers a lot of essential functionalities for programming on GPUs among which:

* OpenGL error detections
* Break points
* Edit and continue on shaders
* FBO[[4]](#fbo) and VBO[[19]](#vbo) inspection at runtime

#### Git

Git is a version control tool, specifically designed to facilitate on one hand the preservation of the historical production and on a second hand, the collaboration between developers on the same project.

## Material environment

### Devlopment environment

#### Modern computers

AGE is designed for modern hardware architectures, it is necessary for its development work on a new computer.

Therefore, the majority of AGE developers have a computer together:

* A multi-core processor (Intel I5 / I7) speeding up the compilation time.
* An SSD hard-drive, bringing Visual Studio 2013 a better response time for auto completion.
* A recent and powerful graphics card for the GPU computations and the display of complex effects.

#### Oculus Rift

The Oculus Rift being integrated into the project, we need to have a Oculus for the AGE development. Therefore, we ordered an Oculus development kit and are awaiting delivery.

The Oculus that we will use will be the last prototype (March 2014). This will allow us to integrate to AGE the latest features offered by the product.

#### Control devices

In order to increase the immersion players, AGE will include the support for the Xbox controller. Therefore the development team of the engine will have in their possession several of these products.

## Technical architecture

### The game engine

#### The Entity Component System architecture

AGE is based on a so-called "Entity Component System Based" architecture. This one is increasingly used in the development of video games and allows the development of any type of gameplay, without any lost in performance.

We will try to explain in the following section the general form of architecture.

##### The entities

Entities are the first objects of the video game, in other words, they represent each separate element in a game (a text, a character, a light, a table, a vase ...).

They are to be understood as containers, initially empty, having only information of a spatial transformation and arranged in a hierarchical tree.

##### The GraphNode, or hierarchical tree

The hierarchical tree, more commonly known as "GraphNode" consists of entities (nodes), its branches representing the relationships between entities. Thus, an entity A can have two child entities (B and C) and a parent entity (Z).

This organization of the entities in a tree allows to operate grouped transformations, for example, a car body having a driver child entity, will make the driver move as the car move.

##### Memory layout

The computer processor is responsible for processing the data, in fact, it applies the calculations asked by the program on a specific memory block.

It is known that the processor can process calculations faster when conducting operations on a contiguous data set in memory (one after the other in the RAM). This is why entities will be grouped in the same memory space of the computer, and the processor will perform the necessary calculations without making any big jump in RAM, thus allowing AGE to offer its users great performances.

##### The components

Entities, as we have explained above, are limited in their definition to be empty containers at first, with only their transformation informations and hierarchical organization.

Components will play a very important role in the specialization of entities.

The components are simple data structure ("plain old data structure"). Each type of component embeds within it a set of data specified by the developer. Thus, there will be components of type "3D Object rendering" which contain all the information required to render 3D object; another one "Point light" will describe the qualities of a light, including its radius, color and intensity.

The components are associated with the entities. Indeed, being contained in an entity, they will participate in the specialization of this one. In other words, the components will define the essence of the entities. Thus, we can imagine an entity with a component "3D Object rendering" and "Point light"; it will then be represented within the game as a 3D object that emits light.

##### The systems

Entities, therefore, act as containers and components define their essence. Systems will dictate the behavior of the entities, depending on the components they contain.

Systems are specialized for one type of entity. Thus, they only deal with entities containing the components they require.

Systems are notified in real time by the engine on each modification of the entities (adding or removing a component). When an entity corresponds to their prerequisites, it is added to their list of items to be processed.

When the system’s update function is called, it performs calculus on each of the entities in the list.

#### Saving and loading

Saving and loading datas is a fundamental requirement of gaming.

Indeed, it should be possible at any time to save on the hard drive the status of a game to be able to load it later, without the player noticing any changes.

Thus, AGE has a loading and saving system. It permit to save the state of each entity and the status of each of its components.

##### Multi-format saving

The saving system will be available in several formats.

First, binary, which will provide developers better performances when writing and reading, as well as better data compression.

However, when the data is written in binary, it is impossible to edit by hand. That is why AGE also offer the ability to save and load data in the JSON format. Indeed, it is text and editing is possible through any text editors.

#### Realist physical simulation

To allow th wider range of gameplay and to provide a more realistic and therefore a better player immersion, AGE propose an integration of realistic physics.

##### Rigid bodies integration

AGE propose to associate entities to physical properties such as collisions shapes.

More commonly called "rigid body", it describe a form from which can be calculated collisions. They can take different forms: sphere, cube, but also much more complex shapes, providing greater accuracy and better realism.

##### Collision informations

Associated with forms of collision, the impact informations will be ​​available for the developers. It may use it to implement different gameplay mechanics. For example, if an entity "shot" collides with a second of type "Character", the developer may, depending on the force of impact change the value of a “life” variable.

##### Character controller

AGE offer a fully integrated "character controller" within its physics engine. The "character controller" allows you to take control over an entity of the game and see it react credibly in his movements. This feature is present in all FPS games (First Person Shooter) which can move in a world. So the character can jump over obstacles above a certain height and cannot pass through walls, for example.

This implementation will allow AGE to provide better realism. Indeed, as we shall see later, the subjective view is at the heart of how the Oculus Rift is meant to be used, and more generally the virtual reality. Therefore, we will offer the possibility of moving into a world from a subjective point of view with physically realistic world, increasing the level of immersion offered by AGE.

#### The asset management

The term "assets" includes all the resources necessary to produce a 3D video game, including the 3D meshes, but also textures, animations or fonts.

AGE offer to the user a wide range of different types of assets, enabling developers to create a rich and full game. Among these assets, AGE offer native integration of these:

* Rendering models
* Textures
* Animations
* Materials (data structure describing the properties for rendering an object)
* Sprites (for integrating 2D animations)
* Fonts
* Collision models (as precalculated collision)

We will see later that these assets will be converted into a specific format using a tool of our design. The conversion of these files will allow a much faster loading and saving.

#### The sound

AGE will offer a full integration of sound to play sound effects and audio tracks.

##### 3D Sound

AGE will incorporate 3D sound management in order to increase the player immersion experience.

##### Sound emitter / Sound receiver

A component type sound emitter is proposed. It will attach a transmitter to its specialized entities in the manner of a speaker.

On the other hand, a component of the type sound receiver will add the option for entities to receive its surrounding sounds.

#### Utilities

A set of utility functions are available to developers to facilitate development. Thus, random generation algorithms and "range" algorithms will be integrated in the engine.

Furthermore, a dependency will register configurations for the product produced with the aid of AGE. These configurations will be editable in the game and automatically saved in a file on the hard disk. In other words, it is possible easily to developers to store the datas required for its application such as screen resolution, sound volume or the input preferences.

#### Benchmarks

As performance is at the heart of the production of video game and interactive 3D experiences, we will set up an benchmark system.

Thus, the developer can evaluate at runtime resource consumption of each part of their program. Including:

• The number of frames per second

• The time taken for rendering

• The number of active entities

• The number of visible entities

• The time consumed by the processor per frame

• The number collision shapes

• The time consumed by each system

• The amount of video memory allocated

* The number of frames per second
* The time taken for rendering
* The number of active entities
* The number of visible entities
* The time consumed by the processor per frame
* The number collision shapes
* The time consumed by each system
* The amount of video memory allocated

### Le moteur graphique

#### Frustrum Culling

Afin d’augmenter les performances du moteur, il est vital d’opérer un culling[[3]](#culling) sur les surfaces non visibles par la caméra.

Ainsi, quel que soit le nombre de polygones dans notre scène le frame rate pourra rester fluide car nous calculerons uniquement l’affichage des objets visibles par la caméra.

#### High Dynamics Range (HDR)

Le HDR est un effet permettant de simuler dynamiquement le fonctionnement de la rétine lors d'un éblouissement. Cet effet est propice à l’immersion du joueur dans des milieux sombre puis lumineux grâce à son impact visuel très réaliste.

#### Glow

Le “Glow” ou “Flou lumineux” est un post-effect (c’est-à-dire pratiqué sur l’image de rendu final) dit “néon” et consiste à flouter la zone entourant l’objet sujet à cette propriété.



Figure 1 : Exemple de Glow

#### Dynamic Shadow Mapping

Technique permettant la mise en place d'ombres réalistes calculées dynamiquement sur tous les types de lumière.

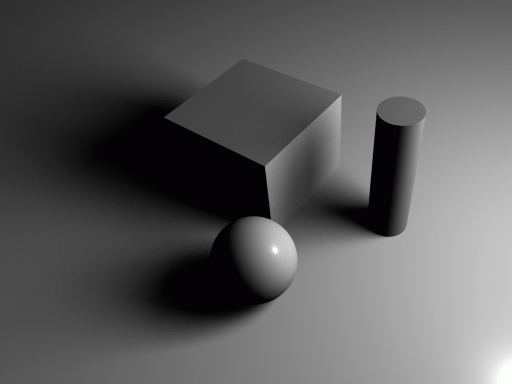


Figure 2 : Dynamic Shadow Mapping

#### Reflexion Mapping

Technique utilisant une Cube Map[[2]](#cubemap) afin de simuler le visuel d'un miroir et cela sur n’importe quel type d’objet.



Figure 3 : Reflexion Mapping sur une sphère

#### Anti-Aliasing

Afin de réduire le crénelage dans nos rendus, nous appliquerons l'algorithme FXAA permettant de diminuer drastiquement la pixellisation des bords de nos géométries.



Figure 4 : Même objet avec à gauche sans FXAA et avec FXAA à droite

#### Tile Based rendering

Le "tile based light culling" est une méthode de culling des lumières qui permet de calculer l'influence de celles-ci uniquement sur les zones de l'écran (Tile) où elles sont nécessaires.

Ceci permettant de réduire considérablement la complexité du calcul d’illumination d’une scène, et ce quel que soit le nombre de lumières dynamiques présentes.

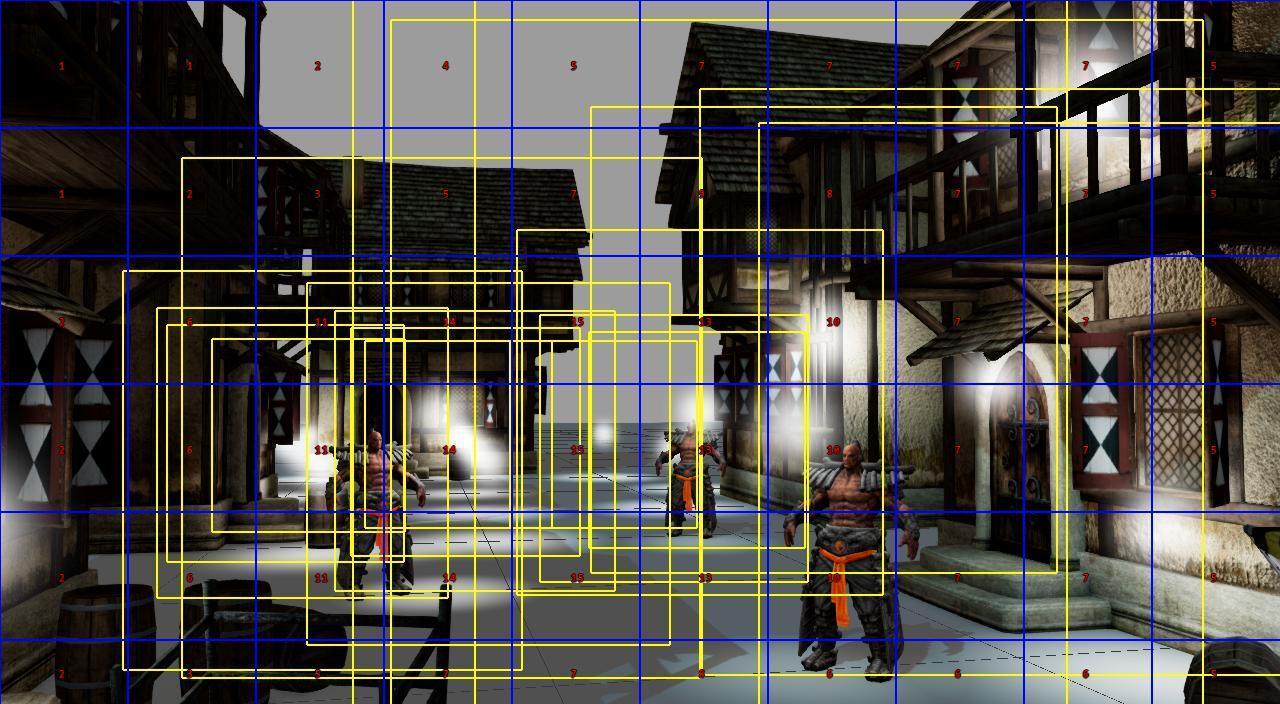


Figure 5 : Effet des lumières sur l'image en fonction des tiles

##### Deffered rendering

Nous allons coupler la technique du "tile based rendering" avec une méthode de rendu appelée "Deffered rendering" permettant le calcul des lumières uniquement sur les pixels réellement affichés.

Ainsi, si un pixel passe le test du light culling, mais ne fait pas partie du rendu final, car il est par exemple caché par une géométrie qui s'affichera par-dessus, les influences des différentes lumières ne seront pas calculées pour celui-ci.

Cet effet permet aussi la mise en place de nombreux algorithmes, notamment certains algorithmes de *global illumination* (simulant l’illumination indirecte dans une scène).

##### Foward rendering

Nous proposerons aussi une méthode de rendu "classique", permettant notamment de rendre des objets semi-transparents dans nos scènes.

Cette méthode va juste rendre tous les objets de la scène, sans tenir compte du fait qu'il soit caché par une quelconque géométrie.

#### Normal mapping

Le Normal mapping est un effet graphique permettant de simuler des reliefs sur des surfaces dans un environnement 3D, et ceci sans avoir à augmenter le nombre de polygones dans la scène.

Le principe de base est d’appliquer une texture sur un objet 3D ayant pour charge de modifier la façon dont elle va être éclairée.

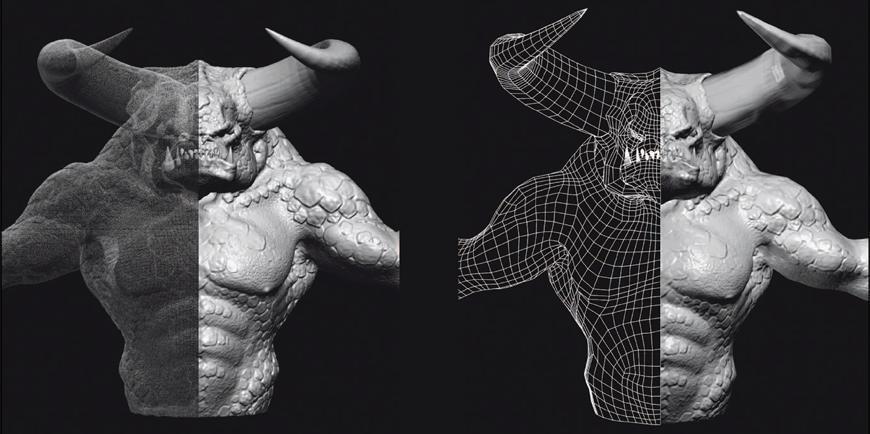


Figure 6: à gauche : rendu obtenu par augmentation du nombre de polygones, à droite : rendu obtenu par normal mapping (même résultat pour des performances accrues).

### Les outils d’éditions

#### Editeur de niveau

Un éditeur de niveau sera mis à la disposition des développeurs. Ce dernier réunira plusieurs fonctionnalités facilitant la production d’un jeu vidéo ou d’une scène 3D.

L’éditeur proposera dans un premier temps de placer l’entité dans l’espace, ainsi que de configurer les différents composants qu’elles contiendront.

Les modifications apportées au sein de l’éditeur de scène pourront être sauvegardées dans un fichier spécifique et chargé au sein d’une partie.

En plus de l’édition de scène, cet éditeur offrira la possibilité de constituer et d’enregistrer des “prefabs”. Ces derniers ne sont rien d’autre qu’une collection d’entités préconfigurées, ce qui facilite par la suite le développement. Ainsi, nous pouvons imaginer un prefab “Ennemi”, lors du déroulement du jeu, il suffira d’appeler la création de ce type de prefab pour faire apparaitre un ennemi dans la scène.

#### Convertisseur d’assets

Un convertisseur d’assets sera proposé dans le cadre de l’utilisation du moteur AGE. Ce dernier proposera de convertir les fichiers natifs, en fichiers compatible avec le moteur.

Lors de cette conversion, les données seront optimisées pour permettre un chargement plus rapide et apporter de meilleures performances au moteur.

Le convertisseur d’assets prendra en entrée différents types de fichiers parmi lesquels : .obj, .fbx, .dae, .tga, .jpg, .png, .mtl, .bullet, .ttf. A partir de ces derniers, il générera de fichiers sérialisés.

De plus le convertisseur d’assets permettra de générer des formes de collision à partir des objets de rendu 3D.

## Composants existants

### Technologies intégrées au moteur

#### SDL

La SDL est une bibliothèque graphique multiplateforme libre et gratuite. Elle est utilisée dans AGE dans le cadre de la création de fenêtres et de la gestion des événements clavier/souris.

#### Bullet

Bullet est un moteur physique permettant la gestion des interactions entre divers corps dans un environnement 3D. Il comprend parmi ses nombreuses fonctionnalités :

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

Tout comme la SDL, il est sous licence libre zlib.

#### FBX SDK

Le SDK FBX est un kit de développement permettant le chargement de modèle FBX. Ce dernier est un format de fichier contenant toutes les informations nécessaires à la réalisation d'objet 3D. Il comprend :

* géométrie
* couleur
* normal
* texture
* animation

Développé par Autodesk il fait partie aujourd'hui des formats les plus utilisés au monde par les graphistes et animateur 3D. Ce SDK nous permet aussi le chargement des modèles 3D de type collada et obj, soit les formats les plus populaires chez les infographistes.

#### Cereal

Cereal est un projet open source donnant accès à plusieurs outils pratiques, à la sérialisation et la désérialisation des structures de données. Nous utilisions ce dernier dans le cadre de la sauvegarde et du chargement des informations.

#### Oculus SDK

L'oculus rift est un casque de réalité augmentée à destination du grand public. Cet outil se présente comme le périphérique central de notre objectif d'immersivité. Ayant pour objectif de renforcer l'expérience de l'utilisateur dans les mondes virtuels, l'oculus rift sera l'occasion pour AGE de mettre en valeur ses effets graphiques et physiques d'une façon innovante et immersive.

#### FMOD

FMOD est une bibliothèque propriétaire utile à la gestion du son 3D. Elle fait partie des outils les plus utilisés dans le monde de la 3D temps réel.

#### GLM

GLM est une bibliothèque de mathématiques conçue pour être utilisée avec OpenGL. Elle nous permet d'effectuer tous nos calculs de transformation.

#### FreeType

FreeType est une bibliothèque open source facilitant le chargement des polices de caractère. Nous nous servons cette dernière dans le cadre de l’asset converter pour générer les images qui seront utilisées pour le texte.

### Technologies de gestion de projet

#### Github

##### Issues – Milestones

Afin de favoriser la communication durant le développement, nous utiliserons le système d'issue[[9]](#issue) et de Milestones[[12]](#milestone) proposé par GitHub. Chaque nouvelle implémentation, mise à jour ou débogage seront précédé par une issue assignée au membre de l'équipe concernée. Ceci permettra une bonne visibilité sur la totalité des tâches à abattre ainsi que de leur réalisation.

Les Milestones permettront de placer ces issues dans un contexte de temps afin d'imposer des échéances aux diverses parties du projet.

##### Pull request

Chaque nouvelle fonctionnalité majeure devra être validée par l'ensemble du groupe avant d'être mergée[[10]](#merge) au cœur du projet, ceci afin de garantir la stabilité du code finale.

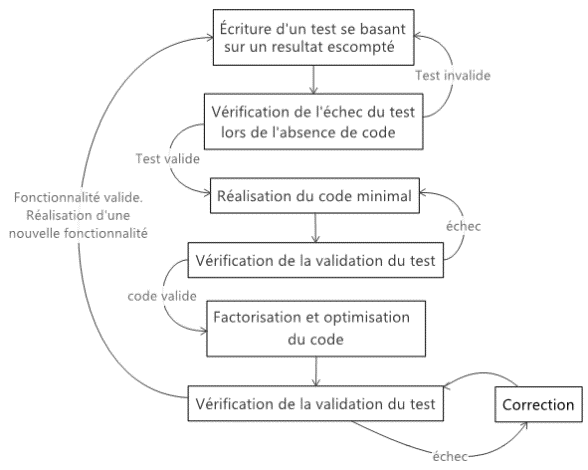
Pour ce faire nous utiliserons le système de pull request[[14]](#pullrequest) et de branche offert par GitHub.

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.

## Stratégie de tests

Le développement du moteur est accompagné d’une stratégie de tests, composée de quatre types de tests permettant à la fois de conserver un code stable, et de répondre aux exigences de nos futurs utilisateurs.

### Tests unitaires

Les tests unitaires garantissent la stabilité du projet, ils seront effectués tout au long du développement et seront développés pour chaque fonction et fonction membre

Nous utiliserons le Google Test Framework, un outil open-source très simple d’utilisation fourni par Google. Nous l’emploierons en appliquant la méthode TDD (Test Driven Development) qui préconise de rédiger nos tests unitaires avant d’écrire le code source de la fonction. Cela permet en premier temps de s’assurer que le code en question est testable unitairement. Le cycle TDD présente en tout quatre à six étapes :

Figure 7: Schéma du cycle TDD

### Tests fonctionnels

Ces tests permettent de s’assurer de la conformité des fonctionnalités vis-à-vis du besoin auquel elles répondent. Pour ce faire, nous allons concevoir et créer quelques jeux utilisant les fonctionnalités à tester pour nous assurer de l’utilisabilité d’AGE. Ils seront faits lorsque le moteur sera suffisamment avancé afin de proposer un maximum de fonctionnalités à tester.

### Tests d’acceptation

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.

### Tests de performance

Afin de tester les performances de notre moteur, nous utiliserons certains programmes permettant d’évaluer l’utilisation du matériel. Niveau CPU, nous utiliserons l’outil de benchmark de Visual Studio 2013. Pour le GPU, nous utiliserons “NSight” (logiciel Nvidia permettant d’évaluer les performances sur des contextes OpenCL et OpenGL), ainsi que Code XL (logiciel AMD) pour avoir une visibilité sur les ressources allouées (Vertex Buffer, Frame Buffer, Texture).

## Communication

Le projet étant un moteur open-source, il est donc important pour nous de développer l’aspect communication dès le début de la réalisation, d’une part pour pouvoir recueillir les différents retours utilisateurs tout au long du développement, et d’autre part pour permettre le support et l'événementiel.

### Identité visuelle

L’identité visuelle est un pilier d’une stratégie de communication. Elle transmet de façon directe une image facilement et rapidement identifiable. Elle permet de transmettre des valeurs et un état d’esprit aux personnes qui pourront se rassembler et adhérer à notre identité. Il est donc important pour nous de créer cette identité, sous forme d’un logotype, et d’une charte graphique pour le site internet.

### Les réseaux sociaux

#### Facebook

Une page dédiée à notre projet va nous permettre de créer et de fédérer une future communauté de testeurs et de créer un lien rapide avec des partenaires. La page Facebook servira essentiellement d’outil de communication social, avec un rythme hebdomadaire de publication sur notre travail, mais aussi de page d'échange sur ce qui touche à notre projet.

#### Twitter

C’est une solution parallèle pour toucher un public plus professionnel et pour communiquer sur des annonces de démos, de concours, ou encore d'évènements au sein d’Epitech (ex: Hackathon). La plateforme sera pour nous un moyen de communiquer de manière ponctuelle. Par exemple, en créant un Hashtag[[7]](#hashtag) pour un événement et en faire la promotion, ou en initiant et encourageant la discussion autour d’AGE pour créer le buzz

#### LinkedIn

L'utilisation de LinkedIn permet d'entretenir les liens entre professionnels, il peut être considéré comme étant un réseau de connaissances. En effet, les avantages dont bénéficient les personnes inscrites sur le réseau sont les échanges d'informations, d'idées et de possibilités. Ces points sont pour nous essentiels ! Investir du temps dans LinkedIn est important puisque c'est un réseau qui est conçu avec l'entreprise.

### Le site

Le cœur de notre communication sera bien entendu notre site web. Le développement du site web comporte de nombreux points pour répondre un maximum à toutes les problématiques de communication que notre projet va devoir affronter.

#### Technologies

Nous utiliserons les technologies Ruby, Javascript et HTML pour créer notre site. Ces technologies sont très axées sur la productivité et sur la simplicité, cela nous permettra de réduire le temps de développement du site pour vraiment débuter le plus tôt possible la communication dans notre projet.

#### Architecture

Le site proposera :

Une page d’accueil pour les informations essentielles de notre projet, les news, les liens importants.

Une rubrique “ Downloads ” pour gérer les différentes ressources qui seront amenées à être partagées avec l’utilisateur du moteur. Bien sûr cette rubrique est amenée à cohabiter avec le dépôt GitHub sur lequel nous travaillerons et qui est aussi une plateforme de ressources. Cette section sera donc réservée aux ressources dites “fixes” ou nécessitant plus de bande passante, pour les assets par exemple.

Un forum. Ce dernier est nécessaire pour deux raisons : la première permet la gestion du support concernant notre moteur. Il est indispensable que les utilisateurs puissent venir poser des questions, partager leurs expériences, et trouver une aide lorsqu’ils en auront besoin. Ensuite, la communication technique sera la principale fonction du forum.

Une rubrique “Support”, cette rubrique est nécessaire pour toute la documentation que nous allons fournir aux utilisateurs du projet. La documentation sera amenée à évoluer au cours du projet, un travail d’actualisation sera donc nécessaire. Une documentation aux objectifs plus axés "recherche et développement", décrira le fonctionnement de certaines implémentations plus complexes comme le "Tile Based Rendering". Afin de pouvoir soumettre aux intéressés un retour d'expérience sur les problèmes rencontrés, les solutions apportés et le résultat obtenus. Cette section est donc une des plus importantes du site en termes de charge de travail.

### Wiki

La création d’un Wiki permet de constituer une base de connaissance et de ressources dynamiques. C'est un espace public collaboratif de travail qui se nourrit des contributions libres du public. Il contiendra un plan détaillé de l’architecture du moteur, ainsi que de ses diverses fonctionnalités et de leur utilisation. Cette documentation a pour objectif d'aider les utilisateurs potentiels à prendre en main le moteur de jeu.

## Organisation de projet

### Une méthode agile : Scrum

L’équipe s’organise à l’aide de méthodes dîtes “agiles”[[11]](#methodesagiles), plus précisément à l’aide de Scrum.

Cette dernière est parfaitement adaptée aux petites équipes, c’est pourquoi nous l’avons choisie pour le développement d’AGE. Elle s’appuie sur le découpage du projet en incréments appelés sprints[[17]](#sprint), le projet grandit en intégrant des fonctionnalités exprimées via la backlog[[1]](#backlog) à l’aide d’un processus itératif.

La méthode Scrum nécessite une forte communication, c’est pourquoi l’équipe se réunira lors d’un scrum[[15]](#scrum) plusieurs fois par semaine pour que chacun exprime l’avancement de sa partie du travail en répondant à ces trois questions :

* Qu'ai-je fait cette semaine ?
* Que dois-je faire la semaine qui vient ?
* Quelles sont les difficultés rencontrées ?

Il existe de plus deux rôles clés pour permettre la mise en place de Scrum :

Le *product owner*, ou propriétaire du produit, désigne le membre de l’équipe qui possède la vision du projet. Il prend les décisions importantes impliquant l’orientation du projet et définit l’ordre dans lequel les fonctionnalités sont développées. Enfin, d’un point de vue technique, il s’assure que la backlog est comprise par toute l’équipe, pour qu’à tout instant, chacun sache parfaitement sur quoi travailler.

Le *scrum master* désigne quant à lui le membre de l’équipe qui est responsable de la méthode. Il a la charge d’optimiser la capacité de production de l’équipe de développement en l’aidant à appliquer la méthode Scrum, à travailler de manière autonome, et à surmonter les obstacles et imprévus qui ralentissent le travail. Il enseigne également au Product Owner la rédaction les différentes parties de la backlog.

Le reste de l’équipe de développement ne comporte pas de rôles prédéfinis, elle est auto-organisée et pluridisciplinaire. Il n’y a pas de hiérarchie interne et toutes les décisions sont prises ensemble.

### Organisation de l’équipe en interne

L’équipe, composée actuellement de sept personnes, s’est organisée de la sorte :

Laureline Patoz

* Scrum Master.
* Développement sur la partie moteur de jeu.
* Développement sur la partie démos, gameplay.

César Leblic

* Product Owner.
* Référent et développeur sur la partie moteur de jeu.
* Développement de la partie Outils

Anthony Boucher

* Développement sur le moteur de jeu.

Paul Baron

* Référent et développeur sur la partie rendu graphique.

Dorian Pinaud

* Référent et développeur sur le pipeline OpenGL.
* Développement de la partie rendu graphique.

Philippe-Henry Sutter

* Communication (événements, site internet, tutoriaux, etc.).
* Développement de la partie moteur de jeu.
* Développement de la partie démos, gameplay.

Théo Bolognini

* Communication (événements, site internet, tutoriaux, etc.)
* Développement de la partie moteur de jeu.
* Développement de la partie démos, gameplay.

# Annexes

## Glossaire

1. *Backlog* - La backlog, ou carnet de produit en français, est une liste de fonctionnalités ou de tâches présentées de manière atomique afin de correspondre aux besoins des utilisateurs. C’est un élément clé de la méthode Scrum. [**↨**](#a_backlog)
2. *Cube map*  - Une texture en forme de cube, c’est à dire 6 faces de taille orientées dans l’espace. [**↨**](#a_cubemap)
3. *Culling* - Le culling est une technique consistant à n’afficher que les polygones visibles. Ainsi les faces cachées ne sont pas calculées.[**↨**](#a_culling)
4. *FBO* - Un FBO ou Frame Buffer Object est un outil OpenGL servant à conserver une frame de rendu, afin de pouvoir y poser certains traitements avant l’affichage final.[**↨**](#a_fbo)
5. *Frame* - Il s’agit d’une image affichée à l’écran durant un temps donné correspondant au rendu d’un moteur graphique. Le frame rate correspond donc à la cadence d’image par seconde. Les jeux vidéo actuels génèrent jusqu’à 200 frames par seconde.[**↨**](#a_frames)
6. *GitHub* - C’est un service d’hébergement et de gestion de développement de logiciel en ligne basé sur le logiciel de gestion de version décentralisé Git.[**↨**](#a_github)
7. *Hashtag* - C’est un marqueur de métadonnées utilisé sur Internet qui permet d’associer un mot-clé à un contenu afin de le partager. Il débute par un croisillon « # » : un hash en anglais.[**↨**](#a_hashtag)
8. *IDE* - Un Integrated Development Environment (IDE) est un environnement de développement regroupant l’ensemble des outils essentiel à la programmation : éditeur de texte, outils de benchmark, compilateur et debugger. [**↨**](#a_ide)
9. *Issue* - Semblable à un ticket, l’issue est sert à représenter un problème qui doit être résolu. Ceci peut-être aussi bien une fonctionnalité à implémenter qu’un bug à supprimer.[**↨**](#a_issue)
10. *Merge* - La fusion [de données en](http://www.linternaute.com/dictionnaire/fr/definition/en/) [un](http://www.linternaute.com/dictionnaire/fr/definition/un/) seul ensemble [selon des](http://www.linternaute.com/dictionnaire/fr/definition/des/) [critères](http://www.linternaute.com/dictionnaire/fr/definition/critere/) de similarité.[**↨**](#a_merge)
11. *Méthodes agiles* - Ce sont des groupes de pratiques de projet apparues en 1991 pouvant s’appliquer à divers types de projet. Elles se veulent plus pragmatiques que les méthodes de développement traditionnel en impliquant au maximum le demandeur et en autorisant les imprévus au cours de la phase de développement.[**↨**](#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* - est une API conçue pour programmer des systèmes parallèles hétérogènes, comme des GPU et des CPU multi-cœur. [**↨**](#a_opencl)
14. *Pull Request*  - Une pull request, ou demande de tirage, Inclusion d’une branche dans la branche mère d’un dépôt. [**↨**](#a_pullrequest)
15. *Un scrum* - Dans la méthode Scrum, un scrum est une réunion régulière et très courte regroupant tous les membres de l’équipe et faisant l’état des lieux des derniers travaux réalisés. [**↨**](#a_scrum)
16. *Shader* - Un shader est un programme informatique qui paramètre une partie du processus de rendu. Ainsi, des effets ou des surfaces qui semblent complexes peuvent être rendus à partir de géométries simples. [**↨**](#a_shaders)
17. *Sprint* - Un sprint est un terme utilisé dans la méthode agile Scrum et représente ces incréments qui composent le projet. Ils peuvent durer entre quelques heures et un mois (avec une préférence pour deux semaines). Chaque sprint commence par une estimation suivie d'une planification opérationnelle. Le sprint se termine par une démonstration de ce qui a été achevé, et contribue à augmenter la valeur d'affaires du produit. Avant de démarrer un nouveau sprint, l'équipe réalise une rétrospective : elle analyse ce qui s'est passé durant ce sprint, afin de s'améliorer pour le prochain. [**↨**](#a_sprint)
18. *Sprite*  - Un sprite désigne une image en deux dimensions qui peut être déplacée, ou qui peut subir d’autres transformations, par rapport à l’écran. [**↨**](#a_sprites)
19. *VBO*  - De l’anglais Vertex Buffer Object, c’est une méthode d’OpenGL qui permet d’envoyer des données 3D vers la carte graphique. Les VBOs permettent de meilleures performances du fait que les données ne sont plus dans la mémoire système, mais résident dans celle du GPU. Avec cette technique, le rendu est donc plus rapide. [**↨**](#a_vbo)