CAHIER DES CHARGES

**MOTEUR DE JEU** **DRAX’ENGINE**

**TEAM** **DRAXTEAM**

Directeur de projet : **teamwar4**

Chef de l'équipe de codage : **kagami**

Vous trouverez en dernière page le récapitulatif des mises à jour.

V0.II.1.5.4

**Sommaire**

Partie I – Présentation 3

I.1 Générale 3

I.1.1 Langage 3

I.1.2 Bibliothèques 3

I.1.3 Compatibilité 3

I.2 Fenêtre 4

I.2.1 Fonctions 4

I.3 Moteur graphique 5

I.3.1 Fonctions 5

I.4 Moteur son 4

I.4.1 Fonctions 3

I.5 Entrées/sorties 4

I.5.1 Fonctions 3

I.6 Moteur réseau 5

I.6.1 Fonctions 3

I.7 Moteur physique 5

I.7.1 Fonctions 3

I.8 Intelligence artificielle 5

I.8.1 Fonctions 3

Partie II – Techniques et codage 6

II.1 Architecture 6

II.1.1 Générale 6

II.1.2 Fenêtre 6

II.1.3 Moteur graphique 7

II.1.4 Moteur son 7

II.1.5 Entrées/sorties 7

II.1.6 Moteur réseau 7

II.1.7 Moteur physique 7

II.1.8 Intelligence artificielle 8

II.2 Répartition 8

II.3 Planning 8

**Partie I – Présentation**

**I.1 Générale**

Le moteur de jeu appelé Drax’Engine sera utilisé dans différents projets en 2D isométrique, aucune source ne sera fournie. Ce dernier sera utilisé uniquement par la Draxteam sous la licence « CC BY NC ND » (Creative Commons). Il comportera un moteur réseau pour pouvoir jouer en multi-joueurs.

***I.1.1 Langage***

Il sera réalisé en C++, car ce langage est puissant et surtout utilise la Programmation Orientée d’Objet. Il est aussi portable et aussi très répandu, ce qui aide pour trouver de la documentation.

***I.1.2 Bibliothèques***

Pour réaliser ce moteur de jeu, nous utiliserons Qt 4.8 et SFML 2.0. Qt servira à la réalisation de fenêtres personnalisées et au moteur réseau tandis que SFML se chargera de la partie graphique et du moteur son. Tous les deux sont puissants et sont compatible avec les principaux systèmes d’exploitation actuels.

***I.1.3 Compatibilité***

Il devra être compatible avec Windows, Linux et Mac. Aucune caractéristique requise n’a été choisi, cependant il devra bien entendu utiliser le moins de ressources possible pour qu’il puisse être lancé sur la plupart des machines.

**I.2 Fenêtre**

Cette partie aura pour but la gestion des fenêtres. Elle sera réalisé presque en totalité avec Qt qui permet la personnalisation de ces dernières.

***I.2.1 Fonctions***

* **Ouvrir/fermer:**

Ce sont les commandes de bases pour une fenêtre, mais toutefois indispensable.

* **Personnalisation :**

Grâce à Qt, il est possible de personnalisé les fenêtres. Il faut que cette possibilité reste accessible que ce soit pour les bordures, les boutons mais aussi pour le nom et l’icône.

* **Changer la résolution :**

A tout moment, le changement de résolution doit être accessible.

* **Passer en plein écran :**

Il en est autant pour le passage en plein écran.

* **Ajouter des widgets Qt :**

Pour créer des interfaces, l’utilisateur doit pouvoir ajouter des widgets et pour les personnaliser.

* **Ajouter une action à un widget :**

L’utilisateur doit être en mesure d’ajouter une action à un widget très facilement.

**I.3 Moteur graphique**

C’est la partie principale du moteur de jeu, c’est elle qui devra afficher le contenu du jeu, pour cela c’est SFML qui sera utilisé.

***I.3.1 Fonctions***

* **Ouvrir/fermer une partie graphique :**

Ce sont les commandes de bases, l’utilisateur doit aussi avoir le choix de la couleur de fond et de la taille.

* **Choix de mode d’affichage :**

Deux modes d’affichages se distinguent, le premier permet l’affichage en extérieur et le second est un affichage intérieur ne représentant seulement une pièce avec seulement deux murs.

|  |  |
| --- | --- |
| **Affichage extérieur** | **Affichage intérieur** |
| * **Afficher la map :**   L’utilisateur devra au préalable choisir la taille et la texture de chaque tiles, ainsi que les objets présents sur la carte. | * **Afficher la map :**   La map ne représentera qu’une seul pièce et seulement les murs arrières seront visibles. |
| * **Ajouter/changer un tile/objet :**   A tout moment, l’utilisateur doit pouvoir ajouter ou changer un tiles ou un objet. | * **Ajouter/changer un tile/objet :**   A tout moment, l’utilisateur doit pouvoir ajouter ou changer un tiles ou un objet. |
| * **Afficher un personnage :**   Permet d’afficher un personnage en fonction des sprites choisis. | * **Afficher un personnage :**   Permet d’afficher un personnage en fonction des sprites choisis. |
| * **Déplacer un personnage :**   Le moteur devra être capable de déplacer le personnage dans huit directions avec une animation. | * **Déplacer un personnage :**   Le moteur devra être capable de déplacer le personnage dans huit directions avec une animation. |
| * **Afficher les lumières :**   Après que l’utilisateur est choisi les différentes options, le moteur devra afficher la lumière. | * **Afficher les lumières :**   Après que l’utilisateur est choisi les différentes options, le moteur devra afficher la lumière et des ombres. |

|  |  |
| --- | --- |
| **Affichage extérieur** | **Affichage intérieur** |
| * **Afficher les ombres :**   Après que l’utilisateur est choisi les différentes options, le moteur devra afficher l’ombre. | * **Afficher les ombres :**   Après que l’utilisateur est choisi les différentes options, le moteur devra afficher l’ombre. |

**I.4 Moteur son**

Cette partie permet d’augmenté la réalité du jeu (bruit de pas, verre cassé,…), mais aussi améliore le confort de jeu.

***I.4.1 Fonctions***

* **Jouer un son :**

Deux sons peuvent être joués en même temps sans que l’un ne coupe l’autre.

* **Jouer une musique :**

Il ne peut y avoir qu’une seul musique à la fois.

**I.5 Entrées/sorties**

Elle permet la gestion des différents périphériques d’un ordinateur, mais aussi celle des différents types de fichiers et bases de données.

***I.5.1 Fonctions***

* **Gestion du clavier :**

Permet de savoir quand une touche du clavier est enfoncée.

* **Gestion de la souris :**

Permet de savoir quand une touche de la souris est enfoncée.

* **Assigner une tache :**

Lorsqu’une touche est enfoncée, le moteur permet d’exécuter la tâche que l’utilisateur a au préalable assignée.

* **Lire/écrire dans un fichier binaire :**

Permet à l’utilisateur d’utiliser les fichiers binaires.

* **Lire/écrire dans un fichier texte:**

Permet à l’utilisateur d’utiliser les fichiers textes.

* **Gestion d’une base de données MySQL:**

Le moteur doit être capable d’envoyer et recevoir des informations provenant d’une base de données MySQL

**I.6 Moteur réseau**

Il permet la gestion de packets de façon sécurisé, c’est grâce à lui que le jeu sera multi-joueurs.

***I.6.1 Fonctions***

* **Gestion des packets:**

Il permet à l’utilisateur d’envoyer et de recevoir des packets de façon sécurisé.

**I.7 Moteur physique**

C’est lui qui s’occupe des calculs, il est une partie très importante du moteur de jeu

***I.7.1 Fonctions***

* **Gestion des collisions:**

Lorsque quelque chose est placé ou déplacé sur la scène, le moteur physique rentre en jeu pour vérifie qu’il y a bien la place pour le mettre.

* **Gestion des ombres:**

Il calcule par rapport à l’intensité, la couleur de la lumière et les différents objets sur la scène, les ombres qui se projettent.

**I.8 Intelligence artificielle**

C’est lui qui décide et calcule les actions que les joueurs ne contrôlent pas.

***I.8.1 Fonctions***

* **Gestion des déplacements des PNJ (personnages non joueur):**

Le moteur doit contrôler les personnages non joueurs et c’est cette partie qui s’occupe de leurs déplacements et de leurs différentes actions.

**Partie II – Techniques et codages**

**II.1 Architecture**

***II.1.1 Générale***