My Project

Generated by Doxygen 1.8.13

Contents

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

Namespace Documentation

4.1 io Namespace Reference

Cet espace sera un espace permettant de définir un buffer custom pour les input, ainsi que de pouvoir afficher tout ce que l'on souhaite.

Functions

- void ChangeTerminal (bool Ech=0)

  Changement des paramètres du terminal.
- void ResetTerminal ()

  Remet le terminal à zero.
- char de ()

  Input.
- void removeLastChar (std::stringstream &i)

  Enlève le dernier caractère d'un stringstream.
- std::string long_input ()

  Long input.
- int getTerminalWidth ()

  Retourne la largeur du terminal.
- int getTerminalHeight ()

  Retourne la hauteur du terminal.
- void bienvenue ()

  Message d'accueil.
- bool checkInput (int x)

  Vérifie que l'user entre des entier.
- competence createCompetence ()

  Creer une competence.
- competence createCompetenceMonstre ()

  Créer une compétence pour monstre (sans mana)
- monstre createMonstre ()

  Créer un monstre.
- std::vector< competence > loadCompetenceFromFile (std::string nomFichier, int numLigne)

  Récupérer les compétences d'un monstre dans le .txt.
- void clearScreen ()

Efface l'écran.

- void afficherCarte (Carte &, int)

  Affichage de la carte.
- void **checkTerminalSize** ()
- template<typename T >
  void afficher (T object)

  Affichage d'objet.
- template<typename T >
  void liste_elements (std::vector< T > vect_element)

  Affichage d'un ensemble d'objets.
- template<typename T >
  T choix_unique_element (std::vector< T > vect_element)

  Choix d'un élément unique.
- template<typename T >
  std::vector< T > **loadAllEntiteFromFile** (T temp, std::string nomFichier)

Variables

- int TermWidth

  Variable retennant la valeur de la largeur de la fenêtre du terminal. Elle permet de réduire le nombre de calculs à faire (étant donné que cette valeur est obtenue avec l'ouverture d'un fichier, son calcul prends donc quelques temps).
- int TermHeight

  Variable retennant la valeur de la hauteur de la fenêtre du terminal. Elle permet de réduire le nombre de calculs à faire (étant donné que cette valeur est obtenue avec l'ouverture d'un fichier, son calcul prends donc quelques temps).
- std::string BLANK

  Chaîne de caractères permettant de remettre à zéro la couleur du texte.
- std::string RED

  Chaîne de caractères permettant de rendre le texte affiché de couleur rouge.
- std::string GREEN

  Chaîne de caractères permettant de rendre le texte affiché de couleur verte.
- std::string YELLOW

  Chaîne de caractères permettant de rendre le texte affiché de couleur jaune.
- std::string BLUE

  Chaîne de caractères permettant de rendre le texte affiché de couleur bleue.
- std::string MAGENTA

  Chaîne de caractères permettant de rendre le texte affiché de couleur magenta.
- int mapPositionX

  Variable permettant de retenir à partir de quelle coordonnée "x" la carte est affichée (si la carte est plus grande que la fenêtre de terminal, cette valeur ne sera pas toujours à 0 ...)
- int mapPositiony

  Variable permettant de retenir à partir de quelle coordonnée "y" la carte est affichée (si la carte est plus grande que la fenêtre de terminal, cette valeur ne sera pas toujours à 0 ...)
- std::pair< int, int > currentPlayerPosition

  Paire de valeurs (std::pair) gardant la position actuelle du joueur dans.

4.1.1 Detailed Description

Cet espace sera un espace permettant de définir un buffer custom pour les input, ainsi que de pouvoir afficher tout ce que l'on souhaite.

4.1.2 Function Documentation

4.1.2.1 afficher()

```
template<typename T >
void io::afficher (
            T object )
```

Affichage d'objet.

Affiche le nom et la description d'un objet.

Parameters

object	Objet à afficher.

4.1.2.2 bienvenue()

```
void io::bienvenue ( )
```

Message d'accueil.

Affiche un message de bienvenue.

4.1.2.3 ChangeTerminal()

```
void io::ChangeTerminal (
            bool Ech = 0 )
```

Changement des paramètres du terminal.

Permet de changer le mode d'entrée de stdin du terminal. Les paramètres présents auparavant sont sauvegardés.

Parameters

Ech	Détermine si on veut que l'entrée utilisateur soit affichée ou pas.

See also

de(), long_input()

4.1.2.4 checkInput()

```
bool io::checkInput (
            int x )
```

Vérifie que l'user entre des entier.

Cette fonction vérifie que l'entrée utilisateur est bien un entier.

Mode opératoire :

- Vérification du failbit de l'entrée utilisateur (std::cin::failbit)
  1. Vidage du buffer
  2. Ignore 256 caractères ou jusqu'a
  3. Affichage d'un message d'erreur d'entrée utilisateur.
  4. Retourne faux
- Sinon retourne vrai

Parameters

x	on sait pas ce qu'il fait là, mais il est là.

4.1.2.5 choix_unique_element()

```
template<typename T >
T io::choix_unique_element (
            std::vector< T > vect_element )
```

Choix d'un élément unique.

Fonction qui prend un vecteur d'éléments en entrée ainsi qu'un booléen, et affiche puis renvoie l'élément choisi.

Parameters

vect_element	Vecteur de l'élément à choisir.
need_desc	Nécessité de description ou non.

Returns

L'élement choisi.

See also

liste_elements(), afficher()

4.1.2.6 createCompetence()

```
competence io::createCompetence ( )
```

Creer une competence.

Cette fonction permet de créer rapidement une compétence pour pouvoir l'utiliser facilement après.

Mode opératoire :

- On crée les variables qui vont tenir les infos rentrées (skillName, skillDamage, skillManaCost)
- On rentre

4.1.2.7 de()

```
char io::de ( )
```

Input.

Gestion des entrées utilisateur, ne prends qu'un seul caractère à la fois.

Voici son mode opératoire :

1. On crée une variable (char)
2. On change la façon dont le terminal gère l'entrée utilisateur avec ChangeTerminal()
3. On utilise la fonction std::getchar() (qui ne prends maintenant qu'un seul caractère sans avoir besoin d'appuyer sur entrée, grâce à ChangeTerminal())
4. On remets les paramètres du terminal comme avant avec ResetTerminal()
5. On retourne l'entrée utilisateur

See also

ChangeTerminal(); ResetTerminal(); long_input()

4.1.2.8 liste_elements()

```
template<typename T >
void io::liste_elements (
            std::vector< T > vect_element )
```

Affichage d'un ensemble d'objets.

Parcourt le vecteur de stockage des objets chargés, et les affiche.

Parameters

vect_element	Vecteur d'éléments.
need_desc	description ou non.

See also

afficher()

4.1.2.9 long_input()

```
std::string io::long_input ( )
```

Long input.

magic.gif

4.1.2.10 removeLastChar()

```
void io::removeLastChar (
            std::stringstream & i )
```

Enlève le dernier caractère d'un stringstream.

Le but de cette fonction est d'enlever le dernier caractère d'un flux de caractères (std::stringstream) étant donné que le C++ ne propose pas de fonction par défaut pour cette fonctionnalité.

Voici son mode opératoire :

1. On prends tout le contenu du stringstream et on le met dans une chaîne de caractères (std::string)
2. Si la chaîne de caractère contient au moins 1 caractère :
   (a) Alors on utilise la fonction std::string::erase(std::string::iterator) pour enlever le dernier caractère
   (b) On remplace le contenu du flux de caractère par du vide
   (c) On remet la chaîne de caractère coupée dans le flux.

Precondition

La fonction recevra un stringstream d'entrée utilisateur. Son but est d'enlever le dernier caractère entré (cette fonction est appelée dans long_input() dans une condition si le caractère rentré est 127, aussi connu sous le nom de DEL ASCII).

Postcondition

La fonction ne retourne rien, car le seul argument est passé par argument (lol) et est donc automatiquement modifié.

Parameters

i	C'est un flux de caractères (std::stringstream) à partir duquel il faudra enlever le dernier caractère.

Chapter 5

Class Documentation

5.1 Carte Class Reference

Public Member Functions

- **Carte** (int taille, std::string name, std::string description)
- void **verif_taille** (int size)
- void **coordonneejoueur** ()
- void **coordonneeobstacle** ()
- void **coordonneemonstre** ()
- void **affichage_normal** ()
- int **nbLigneFichier** (std::string nomFichier)
- void **sauvegarde** ()
- std::string **getName** ()
- std::string **getDescription** ()
- Carte operator= (const Carte &a_copier)

  A déplacer dans fonctionsjeu.h.

Static Public Member Functions

- static std::vector< Carte > **chargement** ()

The documentation for this class was generated from the following file:

- /Users/thibault/GitHub/CERI_software_engineering_game_1/headers/carte.h

5.2 competence Class Reference

Public Member Functions

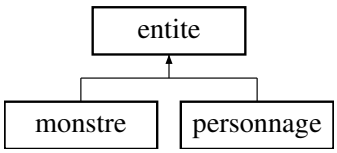
- **competence** (std::string skillName, int skillDamage, int skillManaCost)
- **competence** (std::string skillName, int skillDamage)
- std::string **getName** ()
- std::string **getDescription** ()
- int **getDamage** ()
- int **getManaCost** ()
- template<typename T >
  std::string **toString** (const T &valeur)
- void **printCompetence** ()
- std::string **competenceString** ()

The documentation for this class was generated from the following file:

- /Users/thibault/GitHub/CERI_software_engineering_game_1/headers/competence.h

5.3 entite Class Reference

Inheritance diagram for entite:



Public Member Functions

- entite ()

  Constructeur vide.
- entite (std::string entiteId, std::string entiteName, int entiteHpMax, int entiteSpeed, int entiteManaMax, std::string entiteDescription, std::vector< competence > allSkills)

  Constructeur avec tout.
- template<typename T >
  std::string **toString** (const T &valeur)
- std::string getID ()

  Getter pour l'id.
- std::string getName ()

  Getter pour le nom.
- std::string getDescription ()

  Getter pour la description.
- int getHpMax ()

  Getter pour le nombre de points de vie max.
- int getHpCurrent ()

  Getter pour le nombre de points de vie actuels.

- int getSpeed ()

  Getter pour la vitesse d'attaque de l'entite.
- bool getAlive ()

  Getter qui permet de savoir si l'entite est en vie.
- int getManaMax ()

  Getter pour la mana maximum de l'entite.
- int getManaCurrent ()

  Getter pour la mana actuelle de l'entite.
- std::vector< competence > getSkillVect ()

  Getter qui renvoie un vecteur (std::vector) de compétences.
- int nbLigneFichier (std::string nomFichier)

  Retour d'une string représentant un entite.
- std::string **entiteString** (std::string lettreEntite, std::string nomFichier)
- void saveInFile (std::string lettreEntite, std::string nomFichier)

  Permet d'écrire l'entite dans un fichier de sauvegarde.
- void printEntite ()

  Pour tester.
- bool enleverVie (int degats)

  Enlève x points de vie a l'entite.
- bool enleverMana (int skillManaCost)

  Enlève x points de mana a l'entite.

Protected Attributes

- std::string **entiteId**
- std::string **entiteName**
- std::string **entiteDescription**
- int **entiteHpMax**
- int **entiteHpCurrent**
- int **entiteManaMax**
- int **entiteManaCurrent**
- int **entiteSpeed**
- bool **entiteAlive**
- std::vector< competence > **entiteSkillVect**

5.3.1 Constructor & Destructor Documentation

5.3.1.1 entite()

```
entite::entite (
            std::string entiteId,
            std::string entiteName,
            int entiteHpMax,
            int entiteSpeed,
            int entiteManaMax,
            std::string entiteDescription,
            std::vector< competence > allSkills )
```

Constructeur avec tout.

Parameters

entiteId	L'identifiant de l'entite
entiteName	Le nom de l'entite
entiteHpMax	Les points de vie max de l'entite
entiteSpeed	La vitesse de l'entite
entiteManaMax	Les points de mana max de l'entite
entiteDescription	La description de l'entite
allSkills	Un vecteur (std::vector) contenant toutes les compétences de cette entite.

5.3.2 Member Function Documentation

5.3.2.1 enleverMana()

```
bool entite::enleverMana (
            int skillManaCost )
```

Enlève x points de mana a l'entite.

Cette fonction ne sert à rien, à part ne pas faire bugger les autres.

Returns

Un booléen vérifiant la capacité à dépenser la mana.

5.3.2.2 enleverVie()

```
bool entite::enleverVie (
            int degats )
```

Enlève x points de vie a l'entite.

Cette fonction permet d'enlever des points de vie. Elle permet aussi de savoir si une entite est en vie (ptsVie $<$ 0) ou si elle est morte.

Returns

Un booléen qui est égal à `true` si le entite est mort, `false` sinon.

5.3.2.3 nbLigneFichier()

```
int entite::nbLigneFichier (
            std::string nomFichier )
```

Retour d'une string représentant un entite.

Convertit un objet entite en une ligne de string.

Postcondition

La string contiendra les infos dans cet ordre :

- entiteIdentifiant (type `m<entier>`)
- nom de l'entite
- nombre de points de vie
- vitesse d'attaque
- toutes les compétences , séparées par des **:** Retourne le nombre de lignes d'un fichier.

Compte le nb de lignes du fichier pour créer l'identifiant unique d'un entite. L'identifiant sera `nbLignes + 1`

Returns

Un entier représentant le nombre de lignes.

Parameters

nomFichier	Une string (std::string) qui sera le nom du fichier à ouvrir.

The documentation for this class was generated from the following file:

- /Users/thibault/GitHub/CERI_software_engineering_game_1/headers/entite.h

5.4 jeu Class Reference

Ceci sera la classe du jeu. Elle contient toutes les entités, la carte, ainsi que les fonctions nécessaires à la partie.

`#include <fonctionsjeu.h>`

Public Member Functions

- jeu ()
  Constructeur par défaut sans argument.
- ~jeu ()
  Destructeur par défaut.
- void demarrer_jeu ()
  Fonction permettant de déterminer comment va démarrer la partie.

5.4.1 Detailed Description

Ceci sera la classe du jeu. Elle contient toutes les entités, la carte, ainsi que les fonctions nécessaires à la partie.

Cette classe contient les fonctions nécessaires au démarrage de la partie, au combat, ainsi que toutes les fonctions intermédiaires nécessaires au bon fonctionnement de celles-ci.

Librairies incluses :

- std::stack ,
- io (depuis io.h)

5.4.2 Constructor & Destructor Documentation

5.4.2.1 jeu()

```
jeu::jeu ( )
```

Constructeur par défaut sans argument.

Avec ce constructeur, on peut créer toutes les entités du jeu.

- Chargement de la carte,
- Création d'un personnage,
- Création de tous les monstres.

See also

perso(), carte(), monstre()

5.4.3 Member Function Documentation

5.4.3.1 demarrer_jeu()

```
void jeu::demarrer_jeu ( )
```

Fonction permettant de déterminer comment va démarrer la partie.

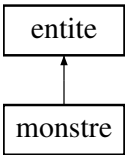
Manque carte pour pouvoir finaliser cette partie.

The documentation for this class was generated from the following file:

- /Users/thibault/GitHub/CERI_software_engineering_game_1/headers/fonctionsjeu.h

5.5 monstre Class Reference

Inheritance diagram for monstre:



Public Member Functions

- monstre ()

  Constructeur vide.
- monstre (std::string entiteId, std::string entiteName, int entiteHpMax, int entiteSpeed, int entiteManaMax, std::string entiteDescription, std::vector< competence > allSkills)

  Constructeur avec tout.
- void printMonstre ()

  Pour tester.

Additional Inherited Members

5.5.1 Constructor & Destructor Documentation

5.5.1.1 monstre()

```
monstre::monstre (
            std::string entiteId,
            std::string entiteName,
            int entiteHpMax,
            int entiteSpeed,
            int entiteManaMax,
            std::string entiteDescription,
            std::vector< competence > allSkills )  [inline]
```

Constructeur avec tout.

Parameters

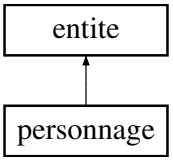
entiteId	L'identifiant du monstre
entiteName	Le nom du monstre
entiteHpMax	Les points de vie max du monstre
entiteSpeed	La vitesse du monstre
entiteManaMax	Les points de mana max du monstre
entiteDescription	La entiteDescription du monstre
allSkills	Un vecteur (std::vector) contenant toutes les compétences de ce monstre.

The documentation for this class was generated from the following file:

- /Users/thibault/GitHub/CERI_software_engineering_game_1/headers/monstre.h

5.6 personnage Class Reference

Inheritance diagram for personnage:



Public Member Functions

- personnage ()

  Constructeur vide.
- **personnage** (std::string entiteId, std::string entiteName, int entiteHpMax, int entiteSpeed, int entiteManaMax, std::string entiteDescription, std::vector< competence > allSkills)
- void printPersonnage ()

  Fonction de test.

Additional Inherited Members

5.6.1 Constructor & Destructor Documentation

5.6.1.1 personnage()

```
personnage::personnage ( )  [inline]
```

Constructeur vide.

Le personnage créé aura 0 de mana, et n'aura aucune description. Mais il sera crée.

The documentation for this class was generated from the following file:

- /Users/thibault/GitHub/CERI_software_engineering_game_1/headers/personnage.h

Index