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.
- void bienvenue ()
  Message d'accueil.
- int getTerminalWidth ()
  Retourne la largeur du terminal.
- int getTerminalHeight ()
  Retourne la hauteur du terminal.
- 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.
- std::vector< monstre > **loadAllMonstreFromFile** ()

- std::vector< personnage > loadAllPersonnageFromFile ()

  Retourne un vecteur contenant tous les personnages du fichier .txt.
- template<typename T >
  void afficher (T object, bool need_desc)

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

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

  Choix d'un élément unique.

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,
            bool need_desc )
```

Affichage d'objet.

Affiche le nom (et éventuellement la description) d'un objet.

Parameters

object	Objet à afficher.
need_desc	Description ou non.

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 la, mais il est la.

4.1.2.5 choix_unique_element()

```
template<typename T >
```

```
T io::choix_unique_element (
            std::vector< T > vect_element,
            bool need_desc )
```

Choix d'un élément unique.

Fonction qui prend un vecteur d'éléments en entrée ainsi qu'un booléen (affichage ou non de la description), 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, damage, manaCost)
- On rentre

4.1.2.7 de()

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

Input.

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

4.1.2.8 liste_elements()

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

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

Chapter 5

Class Documentation

5.1 Carte Class Reference

Public Member Functions

- **Carte** (int taille, std::string name, std::string description)
- void **affichage** ()
- void **sauvegarde** ()
- std::string **getName** ()
- std::string **getDescription** ()
- bool **carte_existe** (std::string nom)
- void **chargement** (std::string nom_selection)
- int **quel_taille** (std::string nom)
- void **suppression** (std::string nom)

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 damage, int manaCost)
- **competence** (std::string skillName, int damage)
- std::string **getSkillName** ()
- 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 ent_combat Struct Reference

Public Member Functions

- void **util_comp** (monstre *entite, competence *comp)

Public Attributes

- monstre * **entite**

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

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

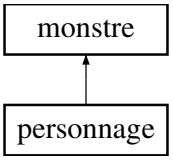
Etant donné que cette fonction utilise des entités externes, il faut que tout le monde aie fini lesdites entités pour que la fonction compile.

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 name, int hpMax, int speed)
  Constructeur avec carac d'un monstre.
- monstre (std::string name, int hpMax, int speed, std::vector< competence > allSkills)
  Constructeur avec les compétences.
- template<typename T >
  std::string **toString** (const T &valeur)
- std::string getName ()
  Getter pour le nom.
- 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 du monstre.
- bool getAlive ()
  Getter qui permet de savoir si le monstre est en vie.
- std::vector< competence > getSkillVect ()
  Getter qui renvoie un vecteur (std::vector) de compétences.
- std::string monstreString ()
  Retour d'une string représentant un monstre.
- int nbLigneFichier (std::string nomFichier)
  Retourne le nombre de lignes d'un fichier.
- void saveInFile ()
  Permet d'écrire le monstre dans un fichier de sauvegarde.
- void printMonstre ()
  Pour tester.
- bool enleverVie (int degats)
  Enlève x points de vie au monstre.

Protected Attributes

- std::string **name**
- int **hpMax**
- int **hpCurrent**
- int **speed**
- bool **alive**
- std::vector< competence > **skillVect**

5.5.1 Constructor & Destructor Documentation

5.5.1.1 monstre() [1/2]

```
monstre::monstre (
            std::string name,
            int hpMax,
            int speed )
```

Constructeur avec carac d'un monstre.

Constructeur assignant tout, sauf les compétences, qui ne seront pas rajoutées (pas de fonctions).

Parameters

name	Le nom du monstre.
hpMax	Le nombre de points de vie max du monstre.
speed	La vitesse d'attaque du monstre.

5.5.1.2 monstre() [2/2]

```
monstre::monstre (
            std::string name,
            int hpMax,
            int speed,
            std::vector< competence > allSkills )
```

Constructeur avec les compétences.

Parameters

name	Le nom du monstre.
hpMax	Le nombre de points de vie max du monstre.
speed	La vitesse d'attaque du monstre.
allSkills	Un vecteur (std::vector) contenant toutes les compétences de ce monstre.

5.5.2 Member Function Documentation

5.5.2.1 enleverVie()

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

Enlève x points de vie au monstre.

Cette fonction permet d'enlever des points de vie. Elle permet aussi de savoir si un monstre est en vie (ptsVie $<$ 0) ou si il est mort.

Returns

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

5.5.2.2 monstreString()

```
std::string monstre::monstreString ( )
```

Retour d'une string représentant un monstre.

Convertit un objet monstre en une ligne de string.

Postcondition

La string contiendra les infos dans cet ordre :

- identifiant (type `m<entier>`)
- nom du monstre
- nombre de points de vie
- vitesse d'attaque
- toutes les compétences , séparées par des `:`

5.5.2.3 nbLigneFichier()

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

Retourne le nombre de lignes d'un fichier.

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

Returns

Un entier représentant le nombre de lignes.

Parameters

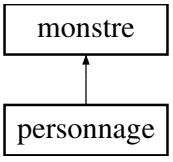
nomFichier	Une string (std::string) qui sera le nom du fichier à ouvrir.

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

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

5.6 personnage Class Reference

Inheritance diagram for personnage:



Public Member Functions

- personnage ()
  Constructeur vide.
- personnage (std::string name, int hpMax, int speed, int manaMax, std::string description)
  Constructeur avec caractéristiques.
- personnage (std::string name, int hpMax, int speed, int manaMax, std::string description, std::vector< competence > allSkills)
  Constructeur avec caractéristiques + vecteur de compétences.
- int getManaMax ()
  Getter pour la mana maximum du personnage.
- int getManaCurrent ()
  Getter pour la mana actuelle du personnage.
- std::string getDescription ()
  Getter pour la desciption du personnage.
- std::string personnageString ()
  Convertit toutes les caracs. d'un personnage en string.
- void savePersoInFile ()
  Ecrit toutes les carac. d'un perso dans un fichier.
- bool enleverMana (int manaCost)
  Return true si le personnage a toujours du mana.
- 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