The Game

Generated by Doxygen 1.8.13

Contents

Chapter 1

Main Page

Bienvenue sur la documentation de The Game, le projet de L2 Informatique (2016/2017).

Vous trouverez ici le documentation (presque) complète du projet.

Author

Raphael Montet, Loïc Menguy, Malvina Gontard, Ludivine Nouveau & Thibault de Villèle.

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

Chapter 4

Class Index

4.1 Class List

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

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

Chapter 6

Namespace Documentation

6.1 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.
- int getTerminalWidth ()

  Retourne la largeur du terminal.
- int getTerminalHeight ()

  Retourne la hauteur du terminal.
- void clearScreen ()

  Efface l'écran.
- void checkTerminalSize ()

  Vérifie la taille du terminal.
- char de ()

  Input.
- std::string long_input ()

  Entrée utilisateur contenant plus d'un caractère.
- void afficherCarte (Carte &, int, bool reset=1)

  Affichage de la carte.
- void updateMap (Carte &jeu_carte, std::pair< int, int > newPlayerPos)

  Met à jour l'affichage de la carte.
- void bienvenue ()

  Message d'accueil.
- void afficherMouvements ()

  Fonction permettant d'afficher un overlay sur la carte.
- void afficherMouvements (std::string erreur_deplacement)

  Fonction permettant d'afficher un overlay sur la carte.
- void afficherMouvements (std::string deplacements_possibles, std::string erreur_deplacement)

Fonction permettant d'afficher un overlay sur la carte.

- void afficherMouvements (std::string message, std::string deplacements_possibles, std::string erreur_↩ deplacement)

  Fonction permettant d'afficher un overlay sur la carte.
- void updateMessage (std::string s, int pos)
- void removeLastChar (std::stringstream &i)

  Enlève le dernier caractère d'un stringstream.
- bool checkInput (int x)

  Vérifie que l'user entre des entiers.
- bool checkSeparatorEntite (std::string uneLigne)

  Verifie qu'une ligne est correcte dans un fichier texte d'entités (bon nombre de séparateurs) + //! Créer une competence.
- bool checkSeparatorSkill (std::string nomFichier, int numLigne)

  Verifie qu'un champ compétences est correct dans un fichier texte d'entités (bon nombre de séparateurs)
- int taille_str (std::string)

  Compte la taille d'une string mieux que la fonction std::string::size(), car elle ne compte pas les accents comme deux caractères.
- void setPlayerPosition (int, int)
- int To_int (std::string input)

  Convertit un string en int.
- 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 >
  void choix_unique_element (T &element, std::vector< T > vect_element, bool combat, bool aff=1)

  Choix d'un élément unique.
- void aff_combat (std::vector< entite > vect_entite)

  Affichage des entités du combat.
- std::vector< competence > loadCompetenceFromFile (std::string nomFichier, int numLigne)

  Chargement des compétences.
- std::vector< Carte > loadAllCarteFromFile (std::string nomFichier)

  Chargement des cartes.
- template<typename T >
  std::vector< T > loadAllEntiteFromFile (T temp, std::string nomFichier)

  Chargement des entités.
- template<typename T >
  std::string toString (const T &valeur)
- bool inputSepCheck (std::string input)

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 ...)
- int interactionsOverlayY

  Stocke la position (x) de l'affichage de l'overlay des actions. Nous n'avons pas besoin du Y car l'overlay prends toute la largeur quoi qu'il arrive.
- std::pair< int, int > currentPlayerPosition

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

6.1.1 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.

6.1.2 Function Documentation

6.1.2.1 aff_combat()

```
void io::aff_combat (
            std::vector< entite > vect_entite )
```

Affichage des entités du combat.

Parcourt le vecteur d'entités, affiche les personnages. Parcourt de nouveau le vecteur, affiche les monstres.

Parameters

vect_entité	Le vecteur d'entités à afficher.

See also

afficher_combat()

6.1.2.2 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.

6.1.2.3 afficherCarte()

```
void io::afficherCarte (
            Carte & ,
            int ,
            bool reset = 1 )
```

Affichage de la carte.

6.1.2.4 afficherMouvements() [1/4]

```
void io::afficherMouvements ( )
```

Fonction permettant d'afficher un overlay sur la carte.

Fonction permettant d'afficher un overlay sur la carte, montrant au joueur dans quelles directions il peut aller. Ne fait qu'appeller afficherMouvements(std::string deplacements_possibles, std::string erreur_deplacement)

See also

afficherMouvements(std::string erreur_deplacement) & afficherMouvements(std::string deplacements_possibles, std::string erreur_deplacement)

6.1.2.5 afficherMouvements() [2/4]

```
void io::afficherMouvements (
            std::string erreur_deplacement )
```

Fonction permettant d'afficher un overlay sur la carte.

Fonction permettant d'afficher un overlay sur la carte, montrant au joueur dans quelles directions il peut aller. Il affiche aussi un message d'erreur si demandé. Ne fait qu'appeller afficherMouvements(std::string deplacements_possibles, std::string erreur_deplacement)

See also

afficherMouvements() & afficherMouvements(std::string deplacements_possibles, std::string erreur_deplacement)

6.1.2.6 afficherMouvements() [3/4]

```
void io::afficherMouvements (
            std::string deplacements_possibles,
            std::string erreur_deplacement )
```

Fonction permettant d'afficher un overlay sur la carte.

Fonction permettant d'afficher un overlay sur la carte, montrant au joueur dans quelles directions il peut aller. Ne fait qu'appeller afficherMouvements(std::string deplacements_possibles, std::string erreur_deplacement)

See also

afficherMouvements() & afficherMouvements(std::string erreur_deplacement)

6.1.2.7 afficherMouvements() [4/4]

```
void io::afficherMouvements (
            std::string message,
            std::string deplacements_possibles,
            std::string erreur_deplacement )
```

Fonction permettant d'afficher un overlay sur la carte.

Fonction permettant d'afficher un overlay sur la carte, montrant au joueur dans quelles directions il peut aller. Ne fait qu'appeller afficherMouvements(std::string deplacements_possibles, std::string erreur_deplacement)

See also

afficherMouvements() & afficherMouvements(std::string erreur_deplacement) & afficherMouvements(std::string deplacements_possibles, std::string erreur_deplacement)

6.1.2.8 bienvenue()

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

Message d'accueil.

Affiche un message de bienvenue.

6.1.2.9 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()

6.1.2.10 checkInput()

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

Vérifie que l'user entre des entiers.

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.

6.1.2.11 checkSeparatorEntite()

```
bool io::checkSeparatorEntite (
            std::string uneLigne )
```

Verifie qu'une ligne est correcte dans un fichier texte d'entités (bon nombre de séparateurs) + //! Créer une competence.

Cette fonction permet de vérifier qu'une ligne contient bien le bon nombre de séparateurs pour éviter les erreurs dans le chargement d'une entité Mode opératoire:

- Parcours de toute la string passée en paramétre
- A chaque séparateur trouvé, on ajoute 1 aux compteurs
- Si le nombre de séparateurs correspond au nombre défini, on retourne true

Parameters

uneLigne	Ligne à vérifier

6.1.2.12 checkSeparatorSkill()

```
bool io::checkSeparatorSkill (
            std::string nomFichier,
            int numLigne )
```

Verifie qu'un champ compétences est correct dans un fichier texte d'entités (bon nombre de séparateurs)

Cette fonction permet de vérifier qu'un champ compétences d'une ligne contient bien le bon nombre de séparateurs pour éviter les erreurs dans le chargement d'une entité Mode opératoire:

- Recherche de la ligne dans le fichier
- Parcours de toute la ligne
- A chaque séparateur trouvé, on ajoute 1 aux compteurs
- Si le nombre de séparateurs correspond au nombre défini, on retourne true

Parameters

nomFichier	Le nom du fichier .txt dans lequel on fait la vérification
numLigne	Le numéro de la ligne à vérifier

6.1.2.13 checkTerminalSize()

```
void io::checkTerminalSize ( )
```

Vérifie la taille du terminal.

6.1.2.14 choix_unique_element()

```
template<typename T >
void io::choix_unique_element (
            T & element,
            std::vector< T > vect_element,
            bool combat,
            bool aff = 1 )
```

Choix d'un élément unique.

Fonction qui prend un élément, un vecteur d'éléments ainsi qu'un booléen en entrée, et affiche les caractéristiques assignées à l'élément.

Parameters

element	Elément dont les caractéristiques doivent être établies.
vect_element	Vecteur de l'élément à choisir.
combat	Situation de combat ou non.

Returns

L'élement choisi.

See also

liste_elements(), afficher(), afficher_detail()

6.1.2.15 clearScreen()

```
void io::clearScreen ( )
```

Efface l'écran.

6.1.2.16 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()

6.1.2.17 getTerminalHeight()

```
int io::getTerminalHeight ( )
```

Retourne la hauteur du terminal.

6.1.2.18 getTerminalWidth()

```
int io::getTerminalWidth ( )
```

Retourne la largeur du terminal.

6.1.2.19 inputSepCheck()

```
bool io::inputSepCheck (
            std::string input )
```

6.1.2.20 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.

See also

afficher()

6.1.2.21 loadAllCarteFromFile()

```
std::vector<Carte> io::loadAllCarteFromFile (
            std::string nomFichier )
```

Chargement des cartes.

Lit toutes les lignes d'un fichier, et remplit un vecteur avec des objets construits à partir des informations récupérées.

Parameters

nomFichier	Le nom du fichier à partir duquel on lit les informations.

Returns

Un vecteur contenant les cartes créées.

6.1.2.22 loadAllEntiteFromFile()

```
template<typename T >
std::vector<T> io::loadAllEntiteFromFile (
            T temp,
            std::string nomFichier )
```

Chargement des entités.

Lit toutes les lignes d'un fichier, et remplit un vecteur avec des objets construits à partir des informations récupérées.

Parameters

temp	Objet dummy permettant au compilateur de comprendre de quel type d'objet il s'agit.
nomFichier	Le nom du fichier à partir duquel on lit les informations.

Returns

Un vecteur contenant les entités créées.

See also

loadCompetenceFromFile()

6.1.2.23 loadCompetenceFromFile()

```
std::vector<competence> io::loadCompetenceFromFile (
            std::string nomFichier,
            int numLigne )
```

Chargement des compétences.

Lit une ligne d'un fichier, et remplit un vecteur avec des objets construits à partir des informations récupérées.

Parameters

nomFichier	Le nom du fichier à partir duquel on lit les informations.
numLigne	La ligne sur laquelle on recherche les informations.

Returns

Un vecteur contenant les compétences créées.

6.1.2.24 long_input()

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

Entrée utilisateur contenant plus d'un caractère.

Mode opératoire :

- Utilise les mêmes fonction de changement du terminal que `de()` (avec la seule différence que l'echo des caractères rentrés est activé), mais possède une boucle qui utilise `getchar()` tant que le caractère entré est différent de la touche `ENTREE`.
- Enlève ensuite le dernier caractère (qui est un caractère `RETOUR_CHARIOT+NEWLINE`).
  - Si la longueur de la chaîne résultante est de 0, alors l'utilisateur n'a rien saisi, donc on lui redemande.
  - Sinon, on renvoie l'entrée utilisateur.

6.1.2.25 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) On enlève le dernier caractère affiché sur stdout (en déplacant le curseur vers la droite après avoir affiché un espace)
   (b) Alors on utilise la fonction std::string::erase(std::string::iterator) pour enlever le dernier caractère
   (c) On remplace le contenu du flux de caractère par du vide
   (d) 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** et est donc automatiquement modifié.

Parameters

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

6.1.2.26 ResetTerminal()

```
void io::ResetTerminal ( )
```

Remet le terminal à zero.

6.1.2.27 setPlayerPosition()

```
void io::setPlayerPosition (
            int ,
            int  )
```

6.1.2.28 taille_str()

```
int io::taille_str (
            std::string  )
```

Compte la taille d'une string mieux que la fonction std::string::size(), car elle ne compte pas les accents comme deux caractères.

6.1.2.29 To_int()

```
int io::To_int (
            std::string input )
```

Convertit un string en int.

Renvoie 0 si l'input n'est pas convertible.

Parameters

input	String à convertir

Returns

Entier obtenu suite à la conversion

6.1.2.30 toString()

```
template<typename T >
std::string io::toString (
             const T & valeur )
```

6.1.2.31 updateMap()

```
void io::updateMap (
             Carte & jeu_carte,
             std::pair< int, int > newPlayerPos )
```

Met à jour l'affichage de la carte.

6.1.2.32 updateMessage()

```
void io::updateMessage (
             std::string s,
             int pos )
```

6.1.3 Variable Documentation

6.1.3.1 BLANK

```
std::string io::BLANK
```

Chaîne de caractères permettant de remettre à zéro la couleur du texte.

6.1.3.2 BLUE

```
std::string io::BLUE
```

Chaîne de caractères permettant de rendre le texte affiché de couleur bleue.

6.1.3.3 currentPlayerPosition

```
std::pair<int,int> io::currentPlayerPosition
```

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

6.1.3.4 GREEN

```
std::string io::GREEN
```

Chaîne de caractères permettant de rendre le texte affiché de couleur verte.

6.1.3.5 interactionsOverlayY

```
int io::interactionsOverlayY
```

Stocke la position (x) de l'affichage de l'overlay des actions. Nous n'avons pas besoin du Y car l'overlay prends toute la largeur quoi qu'il arrive.

6.1.3.6 MAGENTA

```
std::string io::MAGENTA
```

Chaîne de caractères permettant de rendre le texte affiché de couleur magenta.

6.1.3.7 mapPositionX

```
int io::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 ...)

6.1.3.8 mapPositiony

```
int io::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 ...)

6.1.3.9 RED

```
std::string io::RED
```

Chaîne de caractères permettant de rendre le texte affiché de couleur rouge.

6.1.3.10 TermHeight

```
int io::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).

6.1.3.11 TermWidth

```
int io::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).

6.1.3.12 YELLOW

```
std::string io::YELLOW
```

Chaîne de caractères permettant de rendre le texte affiché de couleur jaune.

Chapter 7

Class Documentation

7.1 Carte Class Reference

Classe qui permet de modéliser une carte en mémoire.

`#include <carte.h>`

Public Member Functions

- Carte ()
  Constructeur sans argument.
- Carte (int taille, std::string name, std::string description, int nb_monstre)
  Constructeur avec arguments.
- void coordonneejoueur ()
  Coordonnée Joueur.
- void coordonneeobstacle ()
  Coordonnée obstacle.
- void coordonneemonstre ()
  Coordonnée monstre.
- int nbLigneFichier (std::string nomFichier)
  Nombre de ligne du fichier.
- void saisie ()
  Saisie.
- void saveInFile (std::string lettreCarte, std::string nomFichier)
  Sauvegarde de carte.
- std::string carteString (std::string lettreCarte, std::string nomFichier)
  Carte en ligne.
- std::string getName ()
  Nom.
- std::string getDescription ()
  Description.
- void setTaille (int taille)
  Taille.
- void setId (std::string id)
  Id.

- void setName (std::string name)

  Nom.
- void setDescription (std::string desc)

  Description.
- void setPlateau (int taille)

  Plateau.
- void setCase (int i, int j, std::string value)

  Case.
- void setNbrMonstre (int nbr_monstre)

  Nombre de monstre.
- void setCaseDispo (int case_dispo)

  Case disponible.
- Carte operator= (const Carte a_copier)

  Copie de Carte.
- int getTaille ()

  Taille.
- std::string ** getPlateau ()

  Plateau.
- int getNbrMonstres ()

  Nombre de monstre.
- void affichage_normal ()

  Affichage normal.
- void afficher_detail ()

  Affichage détaillé
- void afficher_brut ()

  Affichage brut.
- bool caseAccessible (int i, int j)

  Case Accessible.
- void echangerContenuCase (int i1, int j1, int i2, int j2)

  Echange le contenu de la case plateau[i1][j1] avec ceux du plateau[i2][j2].
- void monstreMort (int x, int y)

  Fait mourir un monstre à la case x,y et enlève 1 à nbr_monstre.

Static Public Attributes

- static int nbElemProt

  Nombre de cartes protégées.

7.1.1 Detailed Description

Classe qui permet de modéliser une carte en mémoire.

La classe Carte regroupe le plateau de jeu ainsi que le contenu de chaque case et créer une Carte en allocation dynamique.

Chaque Carte est caractérisée par:

un identifiant,

un nom,

une description,

une taille,

un nombre de monstres,

un nombre de cases disponibles,

et un plateau qui lui sont propres.

7.1.2 Constructor & Destructor Documentation

7.1.2.1 Carte() [1/2]

```
Carte::Carte ( )
```

Constructeur sans argument.

Construit une carte en initialisant tous ses paramètres à 0 ou NULL.

7.1.2.2 Carte() [2/2]

```
Carte::Carte (
            int taille,
            std::string name,
            std::string description,
            int nb_monstre )
```

Constructeur avec arguments.

Construit une carte en affectant ses nom, description, et taille, et construit le plateau de jeu à partir de ce dernier paramètre.

Parameters

taille	Taille d'un côté, la carte est carrée
nom	Nom que l'on va donner à la carte
description	Description brève de la carte
nb_monstre	Nombre de monstres présents sur la carte

7.1.3 Member Function Documentation

7.1.3.1 affichage_normal()

```
void Carte::affichage_normal ( )
```

Affichage normal.

Affiche la carte, avec ce que contient chaque case

(affichage du plateau)

7.1.3.2 afficher_brut()

```
void Carte::afficher_brut ( )
```

Affichage brut.

Affiche une phrase avec le nom de la carte, le nombre de cases qu'il y a au total sur la carte et le nombre de monstre qu'il y a exactement sur la carte

7.1.3.3 afficher_detail()

```
void Carte::afficher_detail ( )
```

Affichage détaillé

Affiche un petit texte avec tous les détails de la carte:

Affiche en premier le nom de la carte,

puis la taille d'un coté de la carte, et le nombre de case au total sur la carte

par la suite le nombre de monstre qu'il y a sur la carte,

et enfin affiche la description de la carte

7.1.3.4 carteString()

```
std::string Carte::carteString (
            std::string lettreCarte,
            std::string nomFichier )
```

Carte en ligne.

Convertit toutes les caractéristiques d'une carte en string.

On génère l'ID, les obstacles et les entités sur la carte.

Parameters

lettreCarte	string
nomFichier	string

See also

nbLigneFichier(string)
c_str()
toString(int)

Returns

string ligneFichier

7.1.3.5 caseAccessible()

```
bool Carte::caseAccessible (
            int i,
            int j )
```

Case Accessible.

Vérifie si une case ij est accessible ou pas.

Parameters

i	entier correspondant à la ligne
j	entier correspondant à la colonne

Returns

true si la case ij accessible
false si la case ij n'est pas accessible

7.1.3.6 coordonneejoueur()

```
void Carte::coordonneejoueur ( )
```

Coordonnée Joueur.

On demande les coordonnées du spawn du joueur.

Initialise les coordonnées du spawn de départ du joueur.

On réactualise le nombre de case dispo.

See also

long_input()
c_str()
atoi(string)

7.1.3.7 coordonneemonstre()

```
void Carte::coordonneemonstre ( )
```

Coordonnée monstre.

Demande le nombre de monstres souhaités.

Vérifie que ce nombre est possible en fonction de la taille et du nombre de cases dispo sur la carte.

Demande et initialise les coordonnées de chaque monstre.

Vérifie que la case choisit est libre.

On réactualise le nombre de case dispo.

See also

long_input()
atoi(string)
c_str()

7.1.3.8 coordonneeobstacle()

```
void Carte::coordonneeobstacle ( )
```

Coordonnée obstacle.

Demande le nombre d'obstacles souhaités, vérifie que ce nombre est possible en fonction du nombre de cases dispo sur la carte, demande et initialise les coordonnées de chaque obstacle.

On ne demande pas le genre d'obstacle voulu maintenant.

On réactualise le nombre de case dispo.

See also

long_input()
atoi(string)
c_str()

7.1.3.9 echangerContenuCase()

```
void Carte::echangerContenuCase (
            int i1,
            int j1,
            int i2,
            int j2 )
```

Echange le contenu de la case plateau[i1][j1] avec ceux du plateau[i2][j2].

7.1.3.10 getDescription()

```
std::string Carte::getDescription ( )
```

Description.

Récupère la description de la carte.

string description (de la carte)

7.1.3.11 getName()

```
std::string Carte::getName ( )
```

Nom.

Récupère le nom de la carte.

Returns

string nom (de la carte)

7.1.3.12 getNbrMonstres()

```
int Carte::getNbrMonstres ( )  [inline]
```

Nombre de monstre.

Récupère le nombre de monstre présents sur la carte.

Returns

nbr_monstre int nombre de monstre présent sur la carte

7.1.3.13 getPlateau()

```
std::string** Carte::getPlateau ( )  [inline]
```

Plateau.

Récupère le plateau de la carte.

Returns

plateau string $**$

7.1.3.14 getTaille()

```
int Carte::getTaille ( )  [inline]
```

Taille.

Récupère la taille de la carte.

Returns

taille int

7.1.3.15 monstreMort()

```
void Carte::monstreMort (
            int x,
            int y )
```

Fait mourir un monstre à la case x,y et enlève 1 à nbr_monstre.

7.1.3.16 nbLigneFichier()

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

Nombre de ligne du fichier.

Compte le nb de ligne du fichier pour créer l'identifiant unique d'un monstre.

On ouvre le fichier en lecture, on parcours tout le fichier en incrémentant le compteur à chaque lignes.

Parameters

nomFichier	string

See also

c_str()
ifstream
getline()

Returns

nbLigne : compteur de nombre de ligne.

7.1.3.17 operator=()

```
Carte Carte::operator= (
            const Carte a_copier )
```

Copie de Carte.

On copie une Carte.

On récupère les informations de la carte a_copier pour retourner une nouvelle carte avec ces même caractéristiques.

Parameters

a_copier	Carte

Returns

adresse Carte

7.1.3.18 saisie()

```
void Carte::saisie ( )
```

Saisie.

Permet de créer une carte : initialiser le type d'obstacle et de monstre.

On ouvre le fichier carte, on écrit le nom, la description, la taille et le nombre de monstre qui ont été assigné précédemment.

On parcours ensuite le plateau afin de retrouver la case joueur, les obstacles et de quel type ils sont.

A chaque fois qu'on tombe sur un monstre on récupère le vecteur de monstres pour faire choisir le monstre au client.

See also

ofstream
long_input()
atoi(string)
loadAllEntiteFromFile
afficher_brut()

7.1.3.19 saveInFile()

```
void Carte::saveInFile (
            std::string lettreCarte,
            std::string nomFichier )
```

Sauvegarde de carte.

Ecrit les lignes qui contiennent toutes les caractéristique d'une carte à la fin d'un fichier.

Parameters

lettreCarte	string à écrire
nomfichier	string dans lequel écrire

See also

ofstream
c_str()
carteString(string, string)

7.1.3.20 setCase()

```
void Carte::setCase (
            int i,
            int j,
            std::string value )
```

Case.

Configurer la case de ligne i et de colonne j avec la valeur value. (Joueur, monstres, obstacle).

Parameters

i	int ligne
j	int colonne
value	string valeur de la case

See also

setPlayerPosition(int, int)

7.1.3.21 setCaseDispo()

```
void Carte::setCaseDispo (
            int case_dispo )
```

Case disponible.

Configurer le nombre de case disponibles sur la carte.

Parameters

case_dispo	int nombre de case dispo

7.1.3.22 setDescription()

```
void Carte::setDescription (
            std::string desc )
```

Description.

Configurer la description souhaitée pour la carte.

7.1.3.23 setId()

```
void Carte::setId (
            std::string id )
```

Id.

Configurer l'identifiant de la carte.

Uniquement pour la carte par défaut, pas accessible pour la configuration générale.

7.1.3.24 setName()

```
void Carte::setName (
            std::string name )
```

Nom.

Configurer le nom souhaitée de la carte.

7.1.3.25 setNbrMonstre()

```
void Carte::setNbrMonstre (
            int nbr_monstre )
```

Nombre de monstre.

Configurer le nombre de monstres présent sur la carte.

Parameters

nbr_monstre	int nombre de monstre souhaité

7.1.3.26 setPlateau()

```
void Carte::setPlateau (
            int taille )
```

Plateau.

Configurer la plateau par rapport à la taille demandé. (Suite au chargement d'une carte).

Parameters

taille	int d'un coté de la carte

7.1.3.27 setTaille()

```
void Carte::setTaille (
            int taille )
```

Taille.

Configurer la taille souhaitée de la carte.

7.1.4 Member Data Documentation

7.1.4.1 nbElemProt

```
int Carte::nbElemProt  [static]
```

Nombre de cartes protégées.

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

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

7.2 competence Class Reference

```
#include <competence.h>
```

Public Member Functions

- competence ()

  Constructeur sans arguments.
- competence (std::string skillName, int skillDamage, int skillManaCost)

  Constructeur permettant de créer une compétence pour un personnage.
- competence (std::string skillName, int skillDamage)

  Constructeur permettant de créer une compétence pour un monstre.
- ~competence ()

  Destructeur de compétence.
- std::string getName ()

  Fonction retournant le nom d'une compétence.
- std::string getDescription ()

  Fonction retournant la description d'une compétence.
- int getDamage ()

  Fonction retournant le nombre de points de dégâts à appliquer à une cible.
- int getManaCost ()

  Fonction retournant le nombre de points de magie à utiliser.
- template<typename T >
  std::string toString (const T &valeur)

  Template permettant de retourner une string contenant un élément.
- void afficher_detail ()

  Affichage d'une compétence.
- std::string afficher_detail_combat ()
- std::string competenceString ()

  Conversion en string.

7.2.1 Detailed Description

Permet de contenir une compétence en mémoire.

Une compétence est formée grâce à :

- un nom (`skillName`)
- un nombre de points de dégâts à appliquer (si ce nombre est négatif, la compétence permet de soigner une cible)
- un coût en points de magie (si ce nombre est négatif, la compétence permet de régénerer des points de magie)

NOTE : Par convention, une compétence qui contient un nombre de points de mana sera utilisée sur un joueur. Un monstre n'a pas de magie à gérer.

7.2.2 Constructor & Destructor Documentation

7.2.2.1 competence() [1/3]

```
competence::competence ( )
```

Constructeur sans arguments.

En sortant de ce constructeur, la compétence aura :

- skillName = "Inconnu"
- skillDamage = 0
- skillManaCost = 0

7.2.2.2 competence() [2/3]

```
competence::competence (
            std::string skillName,
            int skillDamage,
            int skillManaCost )
```

Constructeur permettant de créer une compétence pour un personnage.

Les compétences créées avec ce constructeur seront utilisées pour un personnage. La compétence aura donc comme données :

- competence::skillName = skillName
- competence::skillDamage = skillDamage
- competence::skillManaCost = skillManaCost

Parameters

skillName	Nom à assigner à la compétence
skillDamage	Nombre de points de vie à assigner à la compétence (si ce nombre est négatif, la compétence permet de soigner une cible)
skillManaCost	Nombre de points de magie à assigner à la compétence (si ce nombre est négatif, la compétence permet de régénerer des points de magie)

7.2.2.3 competence() [3/3]

```
competence::competence (
            std::string skillName,
            int skillDamage )
```

Constructeur permettant de créer une compétence pour un monstre.

Les compétences créées avec ce constructeur seront utilisées pour un monstre. La compétence aura donc comme données :

- competence::skillName = skillName
- competence::skillDamage = skillDamage

Parameters

skillName	Nom à assigner à la compétence
skillDamage	Nombre de points de vie à assigner à la compétence (si ce nombre est négatif, la compétence permet de soigner une cible)

Note

Les compétences créées utilisant ce constructeur seront assignées à des monstres. Par convention, un monstre n'a pas à gérer de magie.

7.2.2.4 ~competence()

```
competence::~competence ( )
```

Destructeur de compétence.

7.2.3 Member Function Documentation

7.2.3.1 afficher_detail()

```
void competence::afficher_detail ( )
```

Affichage d'une compétence.

Affiche les éléments détaillés d'une compétence (nom, dégâts, coût). Si les dégâts sont négatifs, affiche "soins". Si le coût est négatif, affiche "gain".

7.2.3.2 afficher_detail_combat()

```
std::string competence::afficher_detail_combat ( )
```

7.2.3.3 competenceString()

```
std::string competence::competenceString ( )
```

Conversion en string.

Permet de convertir les caractéristiques d'une compétence en chaîne de caractère en vue de son écriture dans un fichier

Returns

Un string correspondant à la compétence

7.2.3.4 getDamage()

```
int competence::getDamage ( )
```

Fonction retournant le nombre de points de dégâts à appliquer à une cible.

Est utilisée notemment pour l'affichage des possibilités du joueur pendant le combat contre un (des) monstre(s).

Returns

Un entier (`int`) contenant le nombre de points de dégâts à appliquer à une cible.

7.2.3.5 getDescription()

```
std::string competence::getDescription ( )
```

Fonction retournant la description d'une compétence.

Est utilisée notemment pour l'affichage des possibilités du joueur pendant le combat contre un (des) monstre(s).

Returns

Une string (`std::string`) contenant la description.

7.2.3.6 getManaCost()

```
int competence::getManaCost ( )
```

Fonction retournant le nombre de points de magie à utiliser.

Est utilisée notemment pour l'affichage des possibilités du joueur pendant le combat contre un (des) monstre(s).

Returns

Un entier (`int`) contenant le nombre de points de magie à utiliser.

7.2.3.7 getName()

```
std::string competence::getName ( )
```

Fonction retournant le nom d'une compétence.

Est utilisée notemment pour l'affichage des possibilités du joueur pendant le combat contre un (des) monstre(s).

Returns

Une string (`std::string`) contenant le nom.

7.2.3.8 toString()

```
template<typename T >
std::string competence::toString (
            const T & valeur )
```

Template permettant de retourner une string contenant un élément.

Template permettant de retourner une string (`std::string`) contenant un élément de la classe `T` en forme de string.

Returns

Une string (`std::string`) rerésentant l'élément `T`.

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

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

7.3 config Class Reference

```
#include <config.h>
```

Public Member Functions

- void config_carte ()
  Configuration de carte.
- void config_monstre ()
  Configuration de monstre.
- void config_perso ()
  Configuration de personnage.
- template<typename T >
  std::string identif_objet (T objet)
  Caractère d'identification d'un objet.
- int choix_taille ()
  Choix de la taille.
- void creationCarte ()
  Création d'une carte.
- competence createCompetenceEntite (entite dummy, int rang, int manaMax)
  Création de compétence.
- template<class T >
  void creationEntite (T dummy, std::string nom_fichier)
  Création d'une entité
- template<class T >
  void deleteLineElement (std::string nomFichier, std::vector< T > &allElement, std::string lettreElement, int nbElemProt)
  Supprimer un élément.

7.3.1 Member Function Documentation

7.3.1.1 choix_taille()

```
int config::choix_taille ( )
```

Choix de la taille.

Permet de choisir la taille de la carte.

La taille doit être supérieure à 4 et inférieure à 255.

Returns

Taille de la carte

7.3.1.2 config_carte()

```
void config::config_carte ( )
```

Configuration de carte.

Permet de créer ou de supprimer une carte.

7.3.1.3 config_monstre()

```
void config::config_monstre ( )
```

Configuration de monstre.

Permet de créer ou de supprimer un monstre.

7.3.1.4 config_perso()

```
void config::config_perso ( )
```

Configuration de personnage.

Permet de créer ou de supprimer un personnage.

7.3.1.5 createCompetenceEntite()

```
competence config::createCompetenceEntite (
            entite dummy,
            int rang,
            int manaMax )
```

Création de compétence.

Permet de créer une compétence d'entité.

Parameters

dummy	Type spécifique de l'entité à utiliser
rang	Identifiant du numéro de compétence à saisir (parmi le total choisi)

Returns

La compétence créée

7.3.1.6 creationCarte()

```
void config::creationCarte ( )
```

Création d'une carte.

Permet de créer une carte et de la rajouter dans le fichier des cartes.

- Demande un nom pour la carte.
- Demande une description pour la carte.

- Demande une taille pour la carte.
- Construit la carte.
- Demande le placement du joueur.
- Demande le placement des obstacles.
- Demande le placement des monstres.
- Assigne des monstres aux emplacements choisis.
- Ecrit la carte dans le fichier.

7.3.1.7 creationEntite()

```
template<class T >
void config::creationEntite (
            T dummy,
            std::string nom_fichier )  [inline]
```

Création d'une entité

Permet de créer une entité et de la rajouter dans le fichier correspondant.

Parameters

dummy	Type spécifique de l'entité à utiliser
nom_fichier	Nom du fichier dans lequel on veut insérer l'entité

7.3.1.8 deleteLineElement()

```
template<class T >
void config::deleteLineElement (
            std::string nomFichier,
            std::vector< T > & allElement,
            std::string lettreElement,
            int nbElemProt )  [inline]
```

Supprimer un élément.

Cette fonction permet de supprimer un élément choisi par l'utilisateur.

Mode opératoire :

- L'utilisateur choisit l'élément à supprimer dans le vecteur
- On efface du vecteur l'élément choisi
- Effacement de tout le fichier
- Réecriture du fichier via le vecteur d'élément actualisé

Parameters

nomFichier	Le nom du fichier .txt dans lequel on veut supprimer une entité
allEntite	Vecteur contenant tous les éléments disponibles
lettreEntite	String permettant de savoir quel type d'élément on traite
nbElemProt	Entier déterminant le nombre d'éléments par défaut à protéger de la modification

7.3.1.9 identif_objet()

```
template<typename T >
std::string config::identif_objet (
            T objet )  [inline]
```

Caractère d'identification d'un objet.

Permet de récupérer un caractère identifiant un objet.

Parameters

objet	Objet à identifier

Returns

Caractère string identifiant l'objet

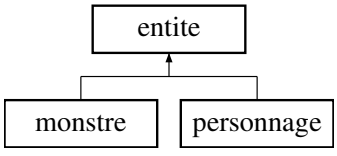
The documentation for this class was generated from the following file:

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

7.4 entite Class Reference

```
#include <entite.h>
```

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.
- 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.
- void setHpCurrent (int current)

  Setter pour points de vie actuels.
- void setManaCurrent (int current)

  Setter pour points de mana actuels.
- int nbLigneFichier (std::string nomFichier)

  Retourne le nombre de lignes d'un fichier.
- std::string entiteString (std::string lettreEntite, std::string nomFichier)

  Convertit une entité en string.
- void saveInFile (std::string lettreEntite, std::string nomFichier)

  Ecriture d'une entité dans un fichier.
- bool is_personnage ()

  Identification personnage.
- bool is_monstre ()

  Identification monstre.
- void afficher_detail ()

  Affichage en détail.
- std::pair< std::string, std::string > afficher_combat ()

  Affichage en combat.
- entite enleverVie (int degats)

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

  Enlève x points de mana a l'entite.
- int randomizeDegat (int damage, int fumbleChance, int critChance)

  Modifie l'application normale des dégâts.
- void afficher_brut ()

  Affichage brut.

Protected Attributes

- std::string entiteId

  Identifiant de l'entité. Est obtenu lors de la lecture du fichier des entités.
- std::string entiteName

  Nom de l'entité.
- std::string entiteDescription

  Description de l'entité.
- int entiteHpMax

  Nombre de points de vie maximale que l'entité peut avoir pendant le jeu.
- int entiteHpCurrent

  Nombre de points de vie que l'entité a au moment "n".
- int entiteManaMax

  Nombre de points de magie maximal que l'entité peut avoir pendant le jeu.
- int entiteManaCurrent

  Nombre de points de magie que l'entité a au moment "n".
- int entiteSpeed

  Vitesse d'attaque de l'entité.
- bool entiteAlive

  Booléen permettant de savoir si l'entité est en vie ou non.
- std::vector< competence > entiteSkillVect

  Vecteur (`std::vector`) permettant de stocker les compétences de l'entité.

7.4.1 Constructor & Destructor Documentation

7.4.1.1 entite() [1/2]

```
entite::entite ( )
```

Constructeur vide.

Crée une entité vide.

Warning

L'entité sera vide. Cela signifie qu'elle ne sera pas utilisable pour le jeu, sa vie étant égale à 0

Postcondition

L'entite crée aura les paramètres suivants:

- entiteName = "Inconnu"
- entiteHpMax = 0
- entiteHpCurrent = 0
- entiteSpeed = 0
- entiteAlive = true (sera changé immédiatement en false)
- entiteSkillVect = <vecteur vide>=""">

7.4.1.2 entite() [2/2]

```
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.

7.4.2 Member Function Documentation

7.4.2.1 afficher_brut()

```
void entite::afficher_brut ( )
```

Affichage brut.

Permet d'afficher les informations nécessaires à la gestion des entités (suppression)

7.4.2.2 afficher_combat()

```
std::pair<std::string,std::string> entite::afficher_combat ( )
```

Affichage en combat.

Permet de limiter l'affichage d'une entité à ses caractéristiques utiles en combat

7.4.2.3 afficher_detail()

```
void entite::afficher_detail ( )
```

Affichage en détail.

7.4.2.4 enleverMana()

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

Enlève x points de mana a l'entite.

Permet de vérifier la possibilité de retirer la mana pour utiliser une compétence, et la retire à l'entité lanceuse si c'est possible.

Parameters

skillManaCost	Coût en mana de la compétence souhaitée

Returns

Un booléen vérifiant la capacité à dépenser la mana.

7.4.2.5 enleverVie()

```
entite 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.

7.4.2.6 entiteString()

```
std::string entite::entiteString (
            std::string lettreEntite,
            std::string nomFichier )
```

Convertit une entité en string.

Permet de prendre une entité, et d'en retourner les informations sous forme de chaîne de caractères

Parameters

lettreEntite	Première lettre du futur identifiant de l'entité
nomFichier	Nom du fichier dans lequel sauvegarder l'entité

Returns

Une chaîne de caractères décrivant l'entité

7.4.2.7 getAlive()

```
bool entite::getAlive ( )
```

Getter qui permet de savoir si l'entite est en vie.

7.4.2.8 getDescription()

```
std::string entite::getDescription ( )
```

Getter pour la description.

7.4.2.9 getHpCurrent()

```
int entite::getHpCurrent ( )
```

Getter pour le nombre de points de vie actuels.

7.4.2.10 getHpMax()

```
int entite::getHpMax ( )
```

Getter pour le nombre de points de vie max.

7.4.2.11 getID()

```
std::string entite::getID ( )
```

Getter pour l'id.

7.4.2.12 getManaCurrent()

```
int entite::getManaCurrent ( )
```

Getter pour la mana actuelle de l'entite.

7.4.2.13 getManaMax()

```
int entite::getManaMax ( )
```

Getter pour la mana maximum de l'entite.

7.4.2.14 getName()

```
std::string entite::getName ( )
```

Getter pour le nom.

7.4.2.15 getSkillVect()

```
std::vector<competence> entite::getSkillVect ( )
```

Getter qui renvoie un vecteur (std::vector) de compétences.

7.4.2.16 getSpeed()

```
int entite::getSpeed ( )
```

Getter pour la vitesse d'attaque de l'entite.

7.4.2.17 is_monstre()

```
bool entite::is_monstre ( )
```

Identification monstre.

Permet de déterminer la qualité de monstre d'une entité.

Returns

Booléen: vrai si l'entité est un monstre, faux sinon

7.4.2.18 is_personnage()

```
bool entite::is_personnage ( )
```

Identification personnage.

Permet de déterminer la qualité de personnage d'une entité.

Returns

Booléen: vrai si l'entité est un personnage, faux sinon

7.4.2.19 nbLigneFichier()

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

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`

Parameters

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

Returns

Un entier représentant le nombre de lignes.

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 **:**

7.4.2.20 randomizeDegat()

```
int entite::randomizeDegat (
            int damage,
            int fumbleChance,
            int critChance )
```

Modifie l'application normale des dégâts.

Permet d'appliquer un modificateur de dégâts à une attaque (néfaste ou bénéfique).

3 cas distincts:

- La valeur aléatoire renvoyée est inférieure au pourcentage plafond de fumble -> les dégâts / soins sont divisés par 2.
- La valeur aléatoire renvoyée est supérieure au pourcentage plancher de critique -> les dégâts / soins sont multipliés par 2.
- La valeur aléatoire renvoyée est entre ces deux limites -> pas de changement.

Parameters

damage	La valeur de dégâts à modifier
fumbleChance	Le pourcentage de chances d'effectuer un fumble (moins de 10% conseillés)
critChance	Le pourcentage de chances d'effectuer un critique

Returns

La valeur modifiée

7.4.2.21 saveInFile()

```
void entite::saveInFile (
            std::string lettreEntite,
            std::string nomFichier )
```

Ecriture d'une entité dans un fichier.

Permet de sauvegarder une entité dans un fichier

Parameters

lettreEntite	Première lettre du futur identifiant de l'entité
nomFichier	Nom du fichier dans lequel sauvegarder l'entité

7.4.2.22 setHpCurrent()

```
void entite::setHpCurrent (
            int current )
```

Setter pour points de vie actuels.

7.4.2.23 setManaCurrent()

```
void entite::setManaCurrent (
            int current )
```

Setter pour points de mana actuels.

7.4.3 Member Data Documentation

7.4.3.1 entiteAlive

```
bool entite::entiteAlive  [protected]
```

Booléen permettant de savoir si l'entité est en vie ou non.

7.4.3.2 entiteDescription

```
std::string entite::entiteDescription  [protected]
```

Description de l'entité.

7.4.3.3 entiteHpCurrent

```
int entite::entiteHpCurrent  [protected]
```

Nombre de points de vie que l'entité a au moment "n".

7.4.3.4 entiteHpMax

```
int entite::entiteHpMax  [protected]
```

Nombre de points de vie maximale que l'entité peut avoir pendant le jeu.

7.4.3.5 entiteId

```
std::string entite::entiteId  [protected]
```

Identifiant de l'entité. Est obtenu lors de la lecture du fichier des entités.

7.4.3.6 entiteManaCurrent

```
int entite::entiteManaCurrent  [protected]
```

Nombre de points de magie que l'entité a au moment "n".

7.4.3.7 entiteManaMax

```
int entite::entiteManaMax  [protected]
```

Nombre de points de magie maximal que l'entité peut avoir pendant le jeu.

7.4.3.8 entiteName

```
std::string entite::entiteName  [protected]
```

Nom de l'entité.

7.4.3.9 entiteSkillVect

```
std::vector<competence> entite::entiteSkillVect  [protected]
```

Vecteur (`std::vector`) permettant de stocker les compétences de l'entité.

7.4.3.10 entiteSpeed

```
int entite::entiteSpeed  [protected]
```

Vitesse d'attaque de l'entité.

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

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

7.5 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 ()

  Temporaire!!
- ~jeu ()

  Destructeur par défaut.
- Carte getCarte ()

  Getter de carte de jeu.
- personnage getPerso ()

  Getter de personnage.
- std::vector< monstre > getMonstres ()

  Getter de vecteur de monstres.
- int getNbMonstres ()

  Getter de nombre de monstres.
- void setJeuCarte (Carte jeu_map)

  Setter de carte de jeu.
- void deplacement (int &result)

  Fonction de déplacement du joueur sur la carte.
- void afficherJeu (int &result)

  Fonction permettant d'afficher la carte, puis de demander un déplacement au joueur.
- std::string genererDeplacement (std::vector< bool > &v)

  Fonction permettant de générer les déplacements possibles à partir d'une case `i, j` du plateau de jeu.
- std::string genererInputAccepte (std::vector< bool > b)

  Fonction permettant de générer une chaine d'entrées utilisateur acceptables pour le déplacement.
- int combat (std::string id_monstre)

  Module de combat.
- std::vector< monstre >::iterator cherche_monstre (std::string id_monstre)

  Recherche de monstre.
- bool chargement_entite (std::vector< entite > &vect_entite, std::string id_monstre)

  Chargement acteurs combat.
- std::vector< int > orga_entites (std::vector< entite > &vect_entite)

  Organisation entités.
- competence choix_comp (entite &indiv)

  Choix compétence.
- entite choix_target (competence comp_util, entite &indiv, std::vector< entite > &vect_entite, std::vector< int > vect_p)

  Choix cible.
- int appliquer_comp (entite indiv, entite target, std::vector< entite > &vect_entite, competence comp_util, int &nb_players, int &nb_monsters)

  Appliquer compétence.
- void quitGame ()

  Quitte le jeu, sans que l'utilisateur n'ai gagné ni perdu.
- void victoireGame ()

  Affiche un message de victoire à l'utilisateur.
- void failedGame ()

  Affiche un message de défait au joueur.

7.5.1 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.

Inclut la librairie io.

7.5.2 Constructor & Destructor Documentation

7.5.2.1 jeu()

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

Temporaire!!

Constructeur par défaut sans argument.

Affichage d'un message de bienvenue. Choix du personnage. Choix de la carte. Chargement des monstres.

See also

perso(), carte(), monstre()

7.5.2.2 ~jeu()

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

Destructeur par défaut.

7.5.3 Member Function Documentation

7.5.3.1 afficherJeu()

```
void jeu::afficherJeu (
            int & result )
```

Fonction permettant d'afficher la carte, puis de demander un déplacement au joueur.

Note

Cette fonction est là uniquement pour des fins de tests. La fonctionnalité qu'elle remplit sera remplacée par d'autres méthodes dans la fichier `tests/main.cpp`.

7.5.3.2 appliquer_comp()

```
int jeu::appliquer_comp (
            entite indiv,
            entite target,
            std::vector< entite > & vect_entite,
            competence comp_util,
            int & nb_players,
            int & nb_monsters )
```

Appliquer compétence.

Permet d'appliquer les effets de la compétence choisie sur la cible choisie. Si la cible meurt, décrémente le compteur de personnages/monstres vivants. Supprime les cibles mortes du vecteur d'entités.

Parameters

target	Cible de la compétence.
vect_entite	Le vecteur duquel on tire la cible de la compétence.
comp_util	La compétence à utiliser.
nb_players	Le nombre total de joueurs de la partie.
nb_monsters	Le nombre de monstres du combat en cours.

Returns

Un entier: 1 si tous les monstres sont morts, 0 si tous les joueurs sont morts, 2 sinon.

See also

enleverVie()

7.5.3.3 chargement_entite()

```
bool jeu::chargement_entite (
            std::vector< entite > & vect_entite,
            std::string id_monstre )
```

Chargement acteurs combat.

Permet de charger tous les acteurs du combat dans un vecteur d'entités.

Parameters

vect_entite	Vecteur où chercher le monstre.
id_monstre	Identifiant du monstre à charger.

Returns

Chargement réussi ou non.

See also

cherche_monstre()

7.5.3.4 cherche_monstre()

```
std::vector<monstre>::iterator jeu::cherche_monstre (
            std::string id_monstre )
```

Recherche de monstre.

Permet de trouver l'objet monstre correspondant à la string id trouvée sur une case. Si la valeur renvoyée correspond à la fin du vecteur, le monstre n'a pas été trouvé.

Parameters

id_monstre	Identifiant du monstre à trouver.

Returns

Un itérateur correspondant à l'élément du vecteur de monstres concerné.

7.5.3.5 choix_comp()

```
competence jeu::choix_comp (
            entite & indiv )
```

Choix compétence.

Permet de sélectionner une compétence par input parmi une liste tirée d'un vecteur (spécifique à chaque entité) Vérifie la possibilité du lancer (niveau de mana). Si l'entité est un monstre, le choix est aléatoire.

Parameters

indiv	L'entité qui joue actuellement.

Returns

Une compétence parmi les compétences utilisables.

See also

choix_unique_element()

7.5.3.6 choix_target()

```
entite jeu::choix_target (
            competence comp_util,
            entite & indiv,
            std::vector< entite > & vect_entite,
            std::vector< int > vect_p )
```

Choix cible.

Permet de choisir une cible parmi une liste tirée d'un vecteur de cibles disponibles. Si l'entité est un monstre, le choix est aléatoire (uniquement parmi les cibles personnages).

Parameters

comp_util	La compétence à utiliser.
indiv	L'entité qui joue actuellement.
vect_entite	Le vecteur duquel on tire la cible de la compétence.
vect_p	Vecteur permettant d'identifier les personnages parmi toutes les entités.

Returns

Une entité, cible de la compétence.

See also

choix_unique_element()

7.5.3.7 combat()

```
int jeu::combat (
            std::string id_monstre )
```

Module de combat.

Permet de gérer le combat.

- Charge les entités (personnages et monstres) contenus dans la case.
- Identifie les personnages et leur nombre.
- Identifie les monstres et leur nombre.
- Pour chaque acteur, choix d'une compétence, puis d'une cible, puis application des effets.

Parameters

id_monstre	Identifiant du monstre à combattre.

Returns

Un entier: 1 si la partie continue, 0 si elle se termine.

See also

chargement_entite(), orga_entites(), aff_combat(), choix_comp(), choix_target(), appliquer_comp()

7.5.3.8 deplacement()

```
void jeu::deplacement (
            int & result )
```

Fonction de déplacement du joueur sur la carte.

Cette fonction permet au joueur de se déplacer sur la carte, en tenant compte des obstacles présents sur ladite carte.

Mode opératoire :

- Génère les déplacements possibles grâce à la fonction `genererDeplacement()` ;
- Génère les entrées utilisateur possibles grâce à la fonction `genererInputAccepte()` ;
- Va chercher la position actuelle du joueur, puis la stocke dans deux entiers (`x` et `y`, oui je sais ces noms sont très originaux) ;
- Afficher les mouvements possibles au joueur grâce à la fonction `afficherMouvements()` ;
- Demande à l'utilisateur où il souhaiterais aller grâce à la fonction `de()`
- Si le joueur rentre un caractère non compris dans la liste des mouvements possibles :
  - On ré-affiche les mouvements possibles avec `afficherMouvements()`, cette fois-ci avec un message d'erreur en plus.
  - On re-demande son choix pour le mouvement grâce à la fonction `de()`
- On change les coordonnées des entiers `x` et `y` en accord avec la demande de l'utilisateur dans un `switch`.
- On met à jour l'affichage de la carte grâce à la fonction `updateMap()`

Postcondition

La position du joueur aura changé. La paire d'entiers `currentPlayerPosition` sera donc mise à jour (grâce à `updateMap()`).

See also

afficherMouvements(), genererDeplacement(), genererInputAccepte(), de() & io::updateMap()

7.5.3.9 failedGame()

```
void jeu::failedGame ( )
```

Affiche un message de défait au joueur.

7.5.3.10 genererDeplacement()

```
std::string jeu::genererDeplacement (
            std::vector< bool > & v )
```

Fonction permettant de générer les déplacements possibles à partir d'une case `i,j` du plateau de jeu.

Cette fonction permet de générer la chaîne de caractères qui affiche les déplacements disponibles au joueur à partir de la case où il se trouve. Mode opératoire :

- On prends les coordonnées actuelles du joueur, que l'on met dans deux entiers créativement appelés `x` et `y`.
- On prends la taille de la carte du jeu grâce à la fonction `carte::getTaille()`. La taille du plateau nous sert à déterminer si une case existe, enlevant ainsi un peu de temps de calcul lors de l'analyse des cases voisines à celle où se trouve le joueur.
- On crée une chaîne de caractères. Cette chaîne servira à stocker les déplacements possibles à afficher par la suite au joueur.
- On vérifie les cases autour du joueur en vérifiant leurs indices et leur contenu (grâce à la fonction `carte↩ ::caseAccessible()`)
- Si la case au dessus du joueur est libre, alors :
  - On ajoute `Z - Haut` à la chaîne de caractères
  - On met à 1 le booléen permettant de savoir si la case est accessible ou non.
- Si la case à la gauche du joueur est libre, alors :
  - On ajoute `Q - Gauche` à la chaîne de caractères
  - On met à 1 le booléen permettant de savoir si la case est accessible ou non.
- Si la case en dessous du joueur est libre, alors :
  - On ajoute `S - Bas` à la chaîne de caractères
  - On met à 1 le booléen permettant de savoir si la case est accessible ou non.
- Si la case à la droite du joueur est libre, alors :
  - On ajoute `D - Droite` à la chaîne de caractères
  - On met à 1 le booléen permettant de savoir si la case est accessible ou non.
- On retourne la châine de caractères générée.

Parameters

v	Vecteur de booléens (`std::vector<bool>`) permettant de savoir quelles cases sont accessibles aux alentours de la case où se trouve le joueur.

Returns

Une châine de caractères à afficher au joueur pour qu'il puisse savoir où il peut aller. La chaine est définie par l'expression régulière suivante : `"|"+(" Z - Haut |")?+(" Q - Gauche |")?+(" S - Bas |")?+(" D - Droite |")?`

Postcondition

La fonction ***ne change absolument rien*** au plateau, ni au jeu. Toutes les données générées pour l'analyse des voisins de la case ont une portée locale.

See also

genererInputAccepte(), deplacement(), carte::caseAccessible() & carte::getTaille()

7.5.3.11 genererInputAccepte()

```
std::string jeu::genererInputAccepte (
            std::vector< bool > b )
```

Fonction permettant de générer une chaine d'entrées utilisateur acceptables pour le déplacement.

Cette fonction permet de générer la chaîne de caractères qui sera analysée pour accepter ou non un déplacement demandé par le joueur à partir de la case où il se trouve.

Mode opératoire :

- Crée une chaîne de caractères (`std::string`) qui contiendra les caractères acceptés lors de l'entrée utilisateur dans la fonction `deplacement()`.
- Lit le vecteur de booléens rempli dans la fonction `genererDeplacement()` :
  - Si le premier booléen est à 1 : on ajoute "Zz" à la chaîne (l'utilisateur pourra donc appuyer sur 'Z' ou 'z' et se déplacer)
  - Si le second booléen est à 1 : on ajoute "Qq" à la chaîne (l'utilisateur pourra donc appuyer sur 'Q' ou 'q' et se déplacer)
  - Si le troisième booléen est à 1 : on ajoute "Ss" à la chaîne (l'utilisateur pourra donc appuyer sur 'S' ou 's' et se déplacer)
  - Si le quatrième booléen est à 1 : on ajoute "Dd" à la chaîne (l'utilisateur pourra donc appuyer sur 'D' ou 'd' et se déplacer)
- Retourne la chaîne de caractères.

Parameters

b	Vecteur de booléens (`std::vector<bool>`) rempli dans la fonction `genererDeplacement()`.

Returns

Une chaine de caractères permettant de déterminer si l'entrée utilisateur est acceptable ou pas. La chaîne est définie par l'expression régulière suivante : `"Zz"?+"Qq"?+"Ss"?+"Dd"?`.

Postcondition

La fonction ***ne change absolument rien*** au plateau, ni au jeu. Mais cette chaîne sera utilisée de la facon suivante : pour déterminer si l'utilisateur a rentré une demande de deplacement valide, on vérifie que le caractère rentré est présent dans la chaîne de caractères générée ici. Si le caractère n'est pas présent, on redemande l'entrée utilisateur au joueur.

See also

genererDeplacement() & deplacements()

7.5.3.12 getCarte()

```
Carte jeu::getCarte ( )
```

Getter de carte de jeu.

7.5.3.13 getMonstres()

```
std::vector<monstre> jeu::getMonstres ( )
```

Getter de vecteur de monstres.

7.5.3.14 getNbMonstres()

```
int jeu::getNbMonstres ( )
```

Getter de nombre de monstres.

7.5.3.15 getPerso()

```
personnage jeu::getPerso ( )
```

Getter de personnage.

7.5.3.16 orga_entites()

```
std::vector<int> jeu::orga_entites (
            std::vector< entite > & vect_entite )
```

Organisation entités.

Permet de trier les entités (selon leur vitesse). Identifie également les indices de vecteur correspondant à des personnages et les stocke dans un vecteur (pour ciblage par monstres).

Parameters

vect_entite	Vecteur de personnages à trier.

Returns

Un vecteur d'entités utilisées pour le combat.

7.5.3.17 quitGame()

```
void jeu::quitGame ( )
```

Quitte le jeu, sans que l'utilisateur n'ai gagné ni perdu.

7.5.3.18 setJeuCarte()

```
void jeu::setJeuCarte (
            Carte jeu_map )
```

Setter de carte de jeu.

7.5.3.19 victoireGame()

```
void jeu::victoireGame ( )
```

Affiche un message de victoire à l'utilisateur.

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

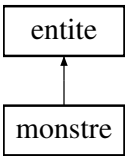
- /Users/thibault/GitHub/CERI_software_engineering_game_1/headers/fonctionsjeu.h

7.6 monstre Class Reference

Classe créant un monstre en mémoire. hérite des propriétés ainsi que des attributs de la classe entite.

```
#include <monstre.h>
```

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 ()

Static Public Attributes

- static int nbElemProt

  Nombre de monstres protégés.

Additional Inherited Members

7.6.1 Detailed Description

Classe créant un monstre en mémoire. hérite des propriétés ainsi que des attributs de la classe entite.

7.6.2 Constructor & Destructor Documentation

7.6.2.1 monstre() [1/2]

```
monstre::monstre ( )
```

Constructeur vide.

Crée un monstre vide.

Warning

Le monstre sera vide. Cela signifie qu'il ne sera pas utilisable pour le jeu, sa vie étant égale à 0

Postcondition

Le monstre crée aura les paramètres suivants:

- entiteName = "Inconnu"
- entiteHpMax = 0
- entiteHpCurrent = 0
- entiteSpeed = 0
- entiteAlive = true (sera changé immédiatement en false)
- entiteSkillVect = <vecteur vide>=""">

7.6.2.2 monstre() [2/2]

```
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 description du monstre
allSkills	Un vecteur (std::vector) contenant toutes les compétences de ce monstre.

7.6.3 Member Function Documentation

7.6.3.1 printMonstre()

```
void monstre::printMonstre ( )
```

7.6.4 Member Data Documentation

7.6.4.1 nbElemProt

```
int monstre::nbElemProt  [static]
```

Nombre de monstres protégés.

Variable de classse permettant d'identifier le nombre de monstres par défaut ne pouvant pas être modifiés (sauf modification directe du code).

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

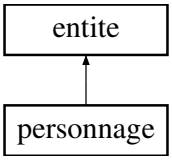
- /Users/thibault/GitHub/CERI_software_engineering_game_1/headers/monstre.h

7.7 personnage Class Reference

Classe personnage héritant de la classe entité

`#include <personnage.h>`

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)
  Constructeur avec arguments.
- void printPersonnage ()
  Fonction de test.

Static Public Attributes

- static int nbElemProt

Additional Inherited Members

7.7.1 Detailed Description

Classe personnage héritant de la classe entité

7.7.2 Constructor & Destructor Documentation

7.7.2.1 personnage() [1/2]

`personnage::personnage ( )`

Constructeur vide.

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

7.7.2.2 personnage() [2/2]

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

Constructeur avec arguments.

Permet de créer un personnage en affectant toutes ses caractéristiques.

Parameters

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

7.7.3 Member Function Documentation

7.7.3.1 printPersonnage()

```
void personnage::printPersonnage ( )
```

Fonction de test.

7.7.4 Member Data Documentation

7.7.4.1 nbElemProt

```
int personnage::nbElemProt  [static]
```

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

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

Chapter 8

File Documentation

8.1 /Users/thibault/GitHub/CERI_software_engineering_game_1/headers/carte.h File Reference

```
#include <iostream>
#include <string>
#include <vector>
```

Classes

- class Carte

  Classe qui permet de modéliser une carte en mémoire.

8.2 /Users/thibault/GitHub/CERI_software_engineering_game_1/headers/competence.h File Reference

```
#include <iostream>
#include <sstream>
#include <string>
```

Classes

- class competence

8.3 /Users/thibault/GitHub/CERI_software_engineering_game_1/headers/config.h File Reference

```
#include "../headers/competence.h"
#include "../headers/io.h"
```

Classes

- class config

Macros

- #define CONFIG_H

8.3.1 Macro Definition Documentation

8.3.1.1 CONFIG_H

```
#define CONFIG_H
```

8.4 /Users/thibault/GitHub/CERI_software_engineering_game_1/headers/entite.h File Reference

```
#include <string>
#include <vector>
#include "competence.h"
```

Classes

- class entite

8.5 /Users/thibault/GitHub/CERI_software_engineering_game_1/headers/fonctionsjeu.h File Reference

```
#include <stack>
#include <vector>
#include "../headers/carte.h"
#include "../headers/competence.h"
#include "../headers/io.h"
#include "../headers/monstre.h"
#include "../headers/personnage.h"
```

Classes

- class jeu

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

Functions

- bool sort_speed (entite a, entite b)

  Tri d'entités.

8.5.1 Function Documentation

8.5.1.1 sort_speed()

```
bool sort_speed (
            entite a,
            entite b )
```

Tri d'entités.

Trie des entités selon la valeur de leur attribut de vitesse.

Parameters

a	Entité par rapport à laquelle on trie.
b	Entité à trier.

Returns

Un booléen: true si la vitesse de a est supérieure à la vitesse de b.

8.6 /Users/thibault/GitHub/CERI_software_engineering_game_1/headers/io.h File Reference

```
#include <algorithm>
#include <fstream>
#include <iostream>
#include <sstream>
#include <stdio.h>
#include <termios.h>
#include <typeinfo>
#include <vector>
#include "../headers/carte.h"
#include "../headers/competence.h"
#include "../headers/entite.h"
#include "../headers/monstre.h"
#include "../headers/personnage.h"
```

Namespaces

- io

  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.

Macros

- #define IO_H

Functions

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

  Changement des paramètres du terminal.
- void io::ResetTerminal ()

  Remet le terminal à zero.
- int io::getTerminalWidth ()

  Retourne la largeur du terminal.
- int io::getTerminalHeight ()

  Retourne la hauteur du terminal.
- void io::clearScreen ()

  Efface l'écran.
- void io::checkTerminalSize ()

  Vérifie la taille du terminal.
- char io::de ()

  Input.
- std::string io::long_input ()

  Entrée utilisateur contenant plus d'un caractère.
- void io::afficherCarte (Carte &, int, bool reset=1)

  Affichage de la carte.
- void io::updateMap (Carte &jeu_carte, std::pair< int, int > newPlayerPos)

  Met à jour l'affichage de la carte.
- void io::bienvenue ()

  Message d'accueil.
- void io::afficherMouvements ()

  Fonction permettant d'afficher un overlay sur la carte.
- void io::afficherMouvements (std::string erreur_deplacement)

  Fonction permettant d'afficher un overlay sur la carte.
- void io::afficherMouvements (std::string deplacements_possibles, std::string erreur_deplacement)

  Fonction permettant d'afficher un overlay sur la carte.
- void io::afficherMouvements (std::string message, std::string deplacements_possibles, std::string erreur_↩ deplacement)

  Fonction permettant d'afficher un overlay sur la carte.
- void io::updateMessage (std::string s, int pos)
- void io::removeLastChar (std::stringstream &i)

  Enlève le dernier caractère d'un stringstream.
- bool io::checkInput (int x)

  Vérifie que l'user entre des entiers.
- bool io::checkSeparatorEntite (std::string uneLigne)

  Verifie qu'une ligne est correcte dans un fichier texte d'entités (bon nombre de séparateurs) + //! Créer une competence.
- bool io::checkSeparatorSkill (std::string nomFichier, int numLigne)

  Verifie qu'un champ compétences est correct dans un fichier texte d'entités (bon nombre de séparateurs)
- int io::taille_str (std::string)

  Compte la taille d'une string mieux que la fonction std::string::size(), car elle ne compte pas les accents comme deux caractères.
- void io::setPlayerPosition (int, int)

- int io::To_int (std::string input)

  Convertit un string en int.
- template<typename T >
  void io::afficher (T object)

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

  Affichage d'un ensemble d'objets.
- template<typename T >
  void io::choix_unique_element (T &element, std::vector< T > vect_element, bool combat, bool aff=1)

  Choix d'un élément unique.
- void io::aff_combat (std::vector< entite > vect_entite)

  Affichage des entités du combat.
- std::vector< competence > io::loadCompetenceFromFile (std::string nomFichier, int numLigne)

  Chargement des compétences.
- std::vector< Carte > io::loadAllCarteFromFile (std::string nomFichier)

  Chargement des cartes.
- template<typename T >
  std::vector< T > io::loadAllEntiteFromFile (T temp, std::string nomFichier)

  Chargement des entités.
- template<typename T >
  std::string io::toString (const T &valeur)
- bool io::inputSepCheck (std::string input)

Variables

- int io::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 io::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 io::BLANK

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

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

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

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

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

  Chaîne de caractères permettant de rendre le texte affiché de couleur magenta.
- int io::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 io::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 ...)
- int io::interactionsOverlayY

  Stocke la position (x) de l'affichage de l'overlay des actions. Nous n'avons pas besoin du Y car l'overlay prends toute la largeur quoi qu'il arrive.
- std::pair< int, int > io::currentPlayerPosition

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

8.6.1 Macro Definition Documentation

8.6.1.1 IO_H

```
#define IO_H
```

8.7 /Users/thibault/GitHub/CERI_software_engineering_game_1/headers/monstre.h File Reference

```
#include "../headers/entite.h"
```

Classes

- class monstre

  Classe créant un monstre en mémoire. hérite des propriétés ainsi que des attributs de la classe entite.

8.8 /Users/thibault/GitHub/CERI_software_engineering_game_1/headers/personnage.h File Reference

```
#include "../headers/entite.h"
```

Classes

- class personnage

  Classe personnage héritant de la classe entité

Index