GolemAdventure

Generated by Doxygen 1.9.8

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

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

Chapter 3

Class Documentation

3.1 arme_s Struct Reference

Public Attributes

- char **nom** [TAILLE_NOM]
- char **desc** [100]
- int **numId**

  numId est l'identifiant de l'objet pour pouvoir équiper le même objet plusieurs fois mais avec un id différent perso↩ Equiper qui sera a 0 s'il n'est pas équipé, sinon 1 pour le 1er perso, 2 pour le 2ème et 3 pour le 3ème dans l'équipe
- perso_t ∗ **persoEquiper**
- typeArmes_t **typeArme**
- attaque_t **att**
- SDL_Texture ∗ **artwork**
- int **cout**

  la valeur d'achat vaut la valeur de vente divisée par 2

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

- lib/type.h

3.2 attaque_s Struct Reference

Public Attributes

- char **nom** [TAILLE_NOM]
- type_att_t **type_att**
- char **desc** [100]
- effet_t **effet**
- int **qte_effet**
- int **duree_effet**
- int **nb_cible**
- int **precision**

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

- lib/type.h

3.3 equipement_s Struct Reference

déclaration du type equipement_t pour donner les stats au stuff

`#include <type.h>`

Public Attributes

- char **nom** [TAILLE_NOM]
- char **desc** [100]
- typeStuff_t **typeEquipement**
- int **numId**

  numId est l'identifiant de l'objet pour pouvoir équiper le même objet plusieurs fois mais avec un id différent persoEquiper qui est un pointeur sur le personnage équipé avec cet objet
- perso_t * **persoEquiper**
- float **Pv**
- float **Mana**
- float **Def**
- float **Res**
- float **For**
- float **Int**
- float **Vit**
- SDL_Texture * **artwork**
- int **cout**

  la valeur d'achat vaut la valeur de vente divisée par 2

3.3.1 Detailed Description

déclaration du type equipement_t pour donner les stats au stuff

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

- lib/type.h

3.4 Image Struct Reference

Structure pour stocker les informations d'une image.

`#include <init_menu.h>`

Public Attributes

- SDL_Texture * **texture**
- SDL_Rect **rect**

3.4.1 Detailed Description

Structure pour stocker les informations d'une image.

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

- lib/init_menu.h

3.5 inventaire_s Struct Reference

déclaration inventaire_t servant à gérer les objets et équipements possédés par le joueur

`#include <type.h>`

Public Attributes

- int **nbObjActu**
- objet_t * **sac** [NB_INV_OBJETS]
- int **nbEquActu**
- equipement_t * **equipements** [NB_INV_EQUIPEMENTS]
- int **nbArmActu**
- arme_t * **armes** [NB_INV_ARMES]
- int **bourse**

3.5.1 Detailed Description

déclaration inventaire_t servant à gérer les objets et équipements possédés par le joueur

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

- lib/type.h

3.6 mob_s Struct Reference

Public Attributes

- char **nom** [TAILLE_NOM]
- int **lvl**
- int **Xp_don**
- float **PvMax**
- float **Pv**
- float **ManaMax**
- float **Mana**
- float **Def**
- float **Res**
- float **For**
- float **Int**
- float **Vit**
- SDL_Texture * **IdleAnim**
- attaque_t **attaques** [NB_ATT]

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

- lib/type.h

3.7 objet_s Struct Reference

déclaration objet_t qui sera pour les potions et objets utilisables

`#include <type.h>`

Public Attributes

- char **nom** [13]
- char **desc** [100]
- float **valPv**
- float **valMp**
- int **cout**
- SDL_Texture * **sprite**

3.7.1 Detailed Description

déclaration objet_t qui sera pour les potions et objets utilisables

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

- lib/type.h

3.8 perso_s Struct Reference

déclaration du type perso pour les stats, animation, etc...

`#include <type.h>`

Public Attributes

- char **Nom** [TAILLE_NOM]
- char **Classe** [20]
- int **lvl**
- int **Xplvl**
- int **Xp**
- int **PtComp**
- float **PvMax**
- float **Pv**
- float **ManaMax**
- float **Mana**
- float **Def**
- float **Res**
- float **For**
- float **Int**
- float **Vit**
- SDL_Texture * **MarcheHori**
- SDL_Texture * **MarcheFront**
- SDL_Texture * **MarcheBack**
- SDL_Texture * **IdleAnim**
- SDL_Texture * **AttAnim**
- equipement_t * **stuff** [TAILLE_EQUIP]
- arme_t * **armes** [NB_ARMES]
- attaque_t **attaques** [NB_ATT]

3.8.1 Detailed Description

déclaration du type perso pour les stats, animation, etc...

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

- lib/type.h

3.9 pnj_s Struct Reference

déclaration du type pnj pour leur texture dialogue etc

```
#include <type.h>
```

Public Attributes

- SDL_Texture * **Dialogue** [6]
- int **NbDialogue**

3.9.1 Detailed Description

déclaration du type pnj pour leur texture dialogue etc

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

- lib/type.h

3.10 team_s Struct Reference

déclaration team_t pour gérer l'équipe des personnages

```
#include <type.h>
```

Public Attributes

- int **nbPerso**
- perso_t * **team** [NB_TEAM_PERSOS]

3.10.1 Detailed Description

déclaration team_t pour gérer l'équipe des personnages

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

- lib/type.h

Chapter 4

File Documentation

4.1 lib/combat.h File Reference

programme gérant les combats.

```
#include <SDL2/SDL.h>
#include <SDL2/SDL_ttf.h>
#include <SDL2/SDL_image.h>
#include <SDL2/SDL_mixer.h>
#include ¨type.h¨
#include ¨texture.h¨
```

Functions

- void DetectEventsFight (int *menu_combat, int *MouseClick, int *MouseOver, int *quit, int *perso, int *obj, scene_t *scene, int w, int h, int *W, int *H, int *att)

  fonction DetectEventsFight qui s'occupera de gérer la détection des événements quand nous sommes en combat
- void ReactEventsFight (int *p, int perso, int obj, int MouseClick, int MouseOver, int *menu_combat, SDL_↩ Renderer *pRenderer, SDL_Texture *fond_combat, int w, int h, inventaire_t *inv, team_t *team, int imob, int H, int W, int att, float statPerso[3][8], scene_t *scene, int *pvMob)

  fonction ReactEventsInvSac qui s'occupe de réagir en fonction des événements quand nous sommes sur le menu d'inventaire sur le sac
- void **fin_combat** (int vict, mob_t *mob, inventaire_t *inv, team_t *team, float statPerso[3][8], scene_t *scene, int *click, int *over)
- int degats_mob (mob_t *mob, attaque_t att, perso_t *perso)

  fonctions de calcul des degats
- int **degats_perso** (perso_t *perso, attaque_t att, mob_t *mob)

  dégâts infligés par les mobs

4.1.1 Detailed Description

programme gérant les combats.

Author

Lenny Borry.

Version

1.9.5

Date

16 Avril 2024.

4.1.2 Function Documentation

4.1.2.1 degats_mob()

```
int degats_mob (
            mob_t * mob,
            attaque_t att,
            perso_t * perso )
```

fonctions de calcul des degats

dégâts infligés par les mobs

4.1.2.2 DetectEventsFight()

```
void DetectEventsFight (
            int * menu_combat,
            int * MouseClick,
            int * MouseOver,
            int * quit,
            int * perso,
            int * obj,
            scene_t * scene,
            int w,
            int h,
            int * W,
            int * H,
            int * att )
```

fonction DetectEventsFight qui s'occupera de gérer la détection des événements quand nous sommes en combat

création de la variable event qui récupère les événements

coordonnés du rectangle d'options en combat

largeur du grand rectangle

largeur des 4 carrés qui coupent le grand rectangle

longueur du rectangle indiquant le nom des persos

largeur du rectangle indiquant le nom des persos

longueur des rectangles ¨attaque¨ et ¨item¨

largeur des rectangles ¨attaque¨ et ¨item¨

longueur des rectangles de chaque attaque

largeur des rectangles de chaque attaque

longueur des rectangles de chaque objet

largeur des rectangles de chaque objet

longueur du rectangle du nom de l'attaque

largeur du rectangle du nom de l'attaque

longueur des rectangles ¨puissance¨ et ¨précision¨

largeur des rectangles ¨puissance¨ et ¨précision¨

taille du carré dans lequel il y aura le logo représentant le type de l'attaque

longueur du rectangle de description de l'attaque

largeur du rectangle de description de l'attaque

longueur du rectangle du nom de l'objet

largeur du rectangle du nom de l'objet

taille du carré du pixel art de l'objet

longueur du rectangle de description de l'objet

largeur du rectangle de description de l'objet

longueur du rectangle du bouton ¨utiliser¨

largeur du rectangle du bouton ¨utiliser¨

écart vertical entre les boutons indiquant le nom des persos

écart horizontal entre les boutons indiquant le nom des persos

écart vertical entre les boutons ¨attaque¨ et ¨item¨

écart horizontal entre les boutons ¨attaque¨ et ¨item¨

écart vertical entre les rectangles de chaque attaque

écart horizontal entre les rectangles de chaque attaque

écart vertical entre les rectangles de chaque objet

écart horizontal entre les rectangles de chaque objet

écart vertical entre les rectangles du nom de l'attaque et de description de l'attaque

écart horizontal entre les rectangles du nom de l'attaque, de la puissance et du logo

écart vertical entre les rectangles ¨puissance¨ et ¨précision¨

écart vertical entre les rectangles du logo et de description

écart vertical entre les rectangles du nom, du pixel art, de la description et du bouton ¨utiliser¨ de l'objet

écart horizontal de chaque côté du rectangle nom de l'objet

écart horizontal de chaque côté du pixel art de l'objet

écart horizontal de chaque côté du rectangle description de l'objet

écart horizontal de chaque côté du bouton ¨utiliser¨

taille du carré du bouton retour

écart entre le bouton retour et le bord

boucle d'attente d'événements

regarde le bouton pour quitter la fenêtre si appuyer met quit à 1 ce qui va arrêter la boucle de jeu

récupération des coordonnées de la souris

regarde si la souris est dans la case des persos

rectangle perso 1

rectangle perso 2

rectangle perso 3

regarde si la souris est dans la case de attaque ou item

rectangle attaque

rectangle item

test si la souris est sur le bouton retour

regarde si la souris est dans la case des attaques

rectangle attaque 1

rectangle attaque 2

rectangle attaque 3

rectangle attaque 4

test si la souris est sur le bouton retour

regarde si la souris est dans la case de description d'attaque

rectangle nom de l'attaque

rectangle description

rectangle puissance

rectangle précision

rectangle description

rectangle du logo

rectangle description

test si la souris est sur le bouton retour

regarde si la souris est dans la case des objets

rectangle objet 1

rectangle objet 2

rectangle objet 3

rectangle objet 4

rectangle objet 5

rectangle objet 6

rectangle objet 7

rectangle objet 8

rectangle objet 9

rectangle objet 10

test si la souris est sur le bouton retour

regarde si la souris est dans la case de description d'objet

rectangle nom de l'objet

rectangle pixel art de l'objet

rectangle description

rectangle utiliser

test si la souris est sur le bouton retour

lancer l'attaque

utiliser l'item

récupération si le bouton de la souris est lâché

regarde si le bouton est relâché sur un des boutons

4.1.2.3 ReactEventsFight()

```
void ReactEventsFight (
            int * p,
            int perso,
            int obj,
            int MouseClick,
            int MouseOver,
            int * menu_combat,
            SDL_Renderer * pRenderer,
            SDL_Texture * fond_combat,
            int w,
            int h,
            inventaire_t * inv,
            team_t * team,
            int imob,
            int H,
            int W,
            int att,
            float statPerso[3][8],
            scene_t * scene,
            int * pvMob )
```

fonction ReactEventsInvSac qui s'occupe de réagir en fonction des événements quand nous sommes sur le menu d'inventaire sur le sac

déclaration de i servant aux boucles for et textwidth, textheight pour les dimensions des textes

longueur du rectangle indiquant le nom des persos

largeur du rectangle indiquant le nom des persos

longueur des rectangles ¨attaque¨ et ¨item¨

largeur des rectangles ¨attaque¨ et ¨item¨

longueur des rectangles de chaque attaque

largeur des rectangles de chaque attaque

longueur des rectangles de chaque objet

largeur des rectangles de chaque objet

longueur du rectangle du nom de l'attaque

largeur du rectangle du nom de l'attaque

longueur des rectangles ¨puissance¨ et ¨précision¨

largeur des rectangles ¨puissance¨ et ¨précision¨

taille du carré dans lequel il y aura le logo représentant le type de l'attaque

longueur du rectangle de description de l'attaque

largeur du rectangle de description de l'attaque

longueur du rectangle du nom de l'objet

largeur du rectangle du nom de l'objet

taille du carré du pixel art de l'objet

longueur du rectangle de description de l'objet

largeur du rectangle de description de l'objet

longueur du rectangle du bouton ¨utiliser¨

largeur du rectangle du bouton ¨utiliser¨

écart vertical entre les boutons indiquant le nom des persos

écart horizontal entre les boutons indiquant le nom des persos

écart vertical entre les boutons ¨attaque¨ et ¨item¨

écart horizontal entre les boutons ¨attaque¨ et ¨item¨

écart vertical entre les rectangles de chaque attaque

écart horizontal entre les rectangles de chaque attaque

écart vertical entre les rectangles de chaque objet

écart horizontal entre les rectangles de chaque objet

écart vertical entre les rectangles du nom de l'attaque et de description de l'attaque

écart horizontal entre les rectangles du nom de l'attaque, de la puissance et du logo

écart vertical entre les rectangles du logo et de description

écart vertical entre les rectangles ¨puissance¨ et ¨précision¨

écart vertical entre les rectangles du nom, du pixel art, de la description et du bouton ¨utiliser¨ de l'objet

écart horizontal de chaque côté du rectangle nom de l'objet

écart horizontal de chaque côté du pixel art de l'objet

écart horizontal de chaque côté du rectangle description de l'objet

écart horizontal de chaque côté du bouton ¨utiliser¨

taille du carré du bouton retour

écart entre le bouton retour et le bord

chaîne de caractères utilisée pour l'affichage des persos avec leur nb de pv et pm

chaîne de caractères pour la conversion d'entiers en caractères

déclaration et calcul des variables des coordonnées des carrés des persos et des mobs

taille du carré du premier perso de la team

taille du carré du 2e perso de la team

taille du carré du 3e perso de la team

écart vertical entre les persos de la team

écart horizontal à gauche du premier perso et à droite du troisieme perso

écart horizontal à droite du premier perso

écart horizontal à gauche et à droite du deuxieme perso

écart horizontal à gauche du troisieme perso

taille du carré du mob

écart vertical au dessus et au dessus du mob

écart horizonral à gauche et à droite du mob

longueur du rectangle du nom des pv

largeur du rectangle du nom des pv

écart horizontal à gauche du 3e perso

écart horizontal à gauche du 2e perso

écart horizontal à gauche du 1er perso

écart vertical au dessus des pv du 3e perso

écart vertical au dessus des pv du 2e perso

écart vertical au dessus des pv du 1er perso

écart vertical entre les pv et le mob

mise en place des coordonnées dans les tableaux correspondants

chargement des textures des boutons de fond et leur version hover (lorsque la souris est dessus)

chargement des textures du bouton retour et de sa version hover (lorsque la souris est dessus)

déclaration des rect correspondant aux coordonnées et dimensions d'affichage de toute l'interface de combat

grand rectangle qui va contenir tous les rectangles ci-dessous

rectangles des persos

rectangles attaque et item

rectangles de chaque objet

rectangles de chaque attaque

rectangles du détail de chaque attaque

rectangles du détail de chaque objet

rectangles de retour

déclaration de la couleur du texte

affichage des fonds à l'écran

déclaration et initialisation des rectangles et des textures de chaque perso et de chaque mob

chaque 'case' représente le n-ième perso. S'il y a 3 persos, je crée le rectangle des 3 persos, s'il y en a 2 j'en crée que 2 etc

de même pour les textures, je charge leur IdleAnim (lorsque le perso bouge sur lui-même)

on se décale d'une image dans le sprite de l'Idle Animation de tous les persos et mobs

déclaration et initialisation des textures des textes de chaque bouton

nom des persos

texture erreur

textures attaque et item

textures de chaque attaque

matrices contenant les 4 attaques de chacun des persos de la team

déclaration et initialisation des textures des objets

lance l'attaque

calcul précision de l'attaque

utilise l'objet

affichage de tous les rectangles selon menu_combat

destruction des textures

délai d'affichage

4.2 combat.h

Go to the documentation of this file.

```
00001
00008 #ifndef COMBAT_H
00009 #define COMBAT_H
00010 #include <SDL2/SDL.h>
00011 #include <SDL2/SDL_ttf.h>
00012 #include <SDL2/SDL_image.h>
00013 #include <SDL2/SDL_mixer.h>
00014 #include "type.h"
00015 #include "texture.h"
00016
00017 void DetectEventsFight(int * menu_combat, int* MouseClick,int *MouseOver, int* quit, int* perso,int*
      obj, scene_t* scene,int w,int h,int * W, int * H,int* att);
00018 void ReactEventsFight(int *p, int perso,int obj, int MouseClick,int MouseOver,int* menu_combat
      ,SDL_Renderer* pRenderer,SDL_Texture* fond_combat,int w,int h,inventaire_t* inv,team_t * team, int
      imob,int H, int W,int att,float statPerso[3][8],scene_t * scene,int* pvMob);
00019 void fin_combat(int vict,mob_t * mob,inventaire_t * inv,team_t* team,float statPerso[3][8],scene_t *
      scene,int* click,int* over);
00020
00021 int degats_mob(mob_t* mob,attaque_t att,perso_t* perso);
00022 int degats_perso(perso_t* perso,attaque_t att,mob_t* mob);
00023
00024 #endif
```

4.3 lib/EventOp.h File Reference

programme gérant les fonctions des événements

```
#include <SDL2/SDL.h>
#include <SDL2/SDL_ttf.h>
#include <SDL2/SDL_image.h>
#include <SDL2/SDL_mixer.h>
#include ¨type.h¨
#include ¨texture.h¨
#include ¨../lib/fmap.h¨
#include ¨../lib/fpnj.h¨
#include ¨../lib/fmobs.h¨
#include ¨../lib/fbibliotheque.h¨
```

Functions

- void transitionFondu (SDL_Renderer *renderer, int largeur_ecran, int hauteur_ecran, int duree)

  déclaration des fonctions DetectEvents pour les sets d'événement possibles selon la scène
- void **chargement** (scene_t *scene, SDL_Renderer *pRenderer, scene_t futurScene)

  fonction faisant l'écran de chargement entre certaines scènes
- void cinematic_start (SDL_Renderer *pRenderer, int *next, SDL_Rect *P1, SDL_Rect *P2, SDL_Rect *P3, SDL_Rect *P4, scene_t *scene, SDL_Texture **storyP1, SDL_Texture **storyP2, SDL_Texture **storyP3, SDL_Texture **storyP4, int *skipOn, SDL_Texture **Map1, SDL_Texture **Map2, SDL_Texture **Map3, SDL_Texture **Map4, SDL_Texture **Map5, SDL_Texture **Map6, int colli[125][250])

  fonction gérant l'affichage et les événements de la cinématique de début de jeu
- void chargement_barre_pv (int, SDL_Texture **, SDL_Texture **, SDL_Renderer *)

  fonction chargement_barre_pv qui s'occupe de charger la bonne texture pour la barre de PV selon le % de PV actuel
- void DetectEventsOp (int *, int *, int *, scene_t *, int, int, int *, int *, int *)

  fonction DetectEventsOp qui s'occupera de gérer la détection des événements quand nous sommes sur l'openWorld
- void DetectEventsInvSac (int *useOk, int *yonUse, int *yonThrow, int *MouseClick, int *MouseOver, scene_t *scene, int w, int h, SDL_Rect destThrowButton)

  fonction DetectEventsInvSac qui s'occupera de gérer la détection des événements quand nous sommes sur le menu d'inventaire sur le sac
- void DetectEventsStats (int *numPage, int *MouseClick, int *MouseOver, scene_t *scene, int w, int h, SDL_Rect *destStatAction, team_t *team, float Stats[3][8])

  fonction DetectEventsStats qui s'occupera de gérer la détection des événements quand nous sommes sur le menu des stats des personnages
- void DetectEventsInvStuff (int *yonThrow, int *MouseClick, int *MouseOver, scene_t *scene, int w, int h, SDL_Rect destThrowButton, int *numPage, int *numPagePerso, int *slot, int *)

  fonction DetectEventsInvStuff qui s'occupera de gérer la détection des événements quand nous sommes sur le menu d'inventaire pour les équipements/armes
- void ReactEventsOp (int *, int, int, int *, SDL_Renderer *, SDL_Texture *, SDL_Rect marcheHori[3][8], SDL_Rect *, int *, int, int, team_t, int biome, int, int *, int[125][250], SDL_Texture *vendeurDialoTexture[5], int *map, SDL_Rect *savedest, int numperso)

  déclaration des fonctions ReactEvents pour les sets d'événement possibles selon la scène
- void ReactEventsInvSac (int *useOk, int *yonUse, int *yonThrow, int MouseClick, int KeyIsPressed, int MouseOver, SDL_Renderer *pRenderer, int w, int h, inventaire_t *, SDL_Rect *destThrowButton, perso_t **team)

  fonction ReactEventsInvSac qui s'occupe de réagir en fonction des événements quand nous sommes sur le menu d'inventaire sur le sac

- void ReactEventsStats (int numPage, int MouseClick, int KeyIsPressed, int MouseOver, SDL_Renderer ∗p↩ Renderer, int w, int h, team_t ∗team, float Stats[3][8], SDL_Rect ∗destStatAction)

  fonction ReactEventsInvSac qui s'occupe de réagir en fonction des événements quand nous sommes sur le menu d'inventaire sur le sac
- void ReactEventsInvStuff (int ∗, int, int, SDL_Renderer ∗, int, int, inventaire_t ∗, SDL_Rect ∗, team_t ∗, int, int, int, int ∗)

  fonction ReactEventsInvStuff qui s'occupe de réagir en fonction des événements quand nous sommes sur le menu d'inventaire pour les équipements/armes
- void EventShop (int ∗MouseOver, int ∗Action, SDL_Renderer ∗pRenderer, int biome, equipement_t all↩ Stuff[ALL_EQUIPEMENTS][3], inventaire_t ∗inv, int w, int h, scene_t ∗scene, objet_t allPot[ALL_OBJETS], int ∗, arme_t armes[ALL_ARMES][3])

  fonction EventsInvShop qui gère le menu d'achat dans le magasin pour acheter équipement, armes et objets
- void EventSell (int ∗MouseOver, int ∗MouseClick, int ∗Action, SDL_Renderer ∗pRenderer, inventaire_t ∗inv, int w, int h, scene_t ∗scene, int ∗numPage)
- int achat_possible (int choiceBiy, int biome, inventaire_t ∗inv, objet_t allPot[ALL_OBJETS], equipement_t allStuff[ALL_EQUIPEMENTS][3], arme_t armes[ALL_ARMES][3])

  vérifie si on a assez d'argent pour acheter l'item sélectionné, si on a la place pour le stocker ou si on n'a pas 3 fois le même équipement si les conditions sont réunies, on ne touche pas à achatPossible, sinon si l'une des vérifications ne vaut pas vrai on met achatPossible à 0

4.3.1 Detailed Description

programme gérant les fonctions des événements

Author

Luka Cognard.

Version

1.9.5

Date

16 Avril 2024.

4.3.2 Function Documentation

4.3.2.1 achat_possible()

```
int achat_possible (
            int choiceBiy,
            int biome,
            inventaire_t * inv,
            objet_t allPot[ALL_OBJETS],
            equipement_t allStuff[ALL_EQUIPEMENTS][3],
            arme_t armes[ALL_ARMES][3] )
```

vérifie si on a assez d'argent pour acheter l'item sélectionné, si on a la place pour le stocker ou si on n'a pas 3 fois le même équipement si les conditions sont réunies, on ne touche pas à achatPossible, sinon si l'une des vérifications ne vaut pas vrai on met achatPossible à 0

cas où magasin du biome 1 à 3

cas où magasin du biome 5

cas où magasin du biome 4

cas où magasin du biome 1 à 3

cas où magasin du biome 5

cas où magasin du biome 4

objets

cas différents selon chaque biome

biome 1

biome 2

biome 3

biome 4

biome 5

4.3.2.2 chargement_barre_pv()

```
void chargement_barre_pv (
            int pourcent,
            SDL_Texture ** pvJauge,
            SDL_Texture ** pvActu,
            SDL_Renderer * pRenderer )
```

fonction chargement_barre_pv qui s'occupe de charger la bonne texture pour la barre de PV selon le % de PV actuel

barre verte si le nombre de PV est supérieur ou égal à 50%

barre jaune si le nombre de PV est supérieur ou égal à 20% mais est inférieur à 50%

barre rouge si le nombre de PV est inférieur à 20%

4.3.2.3 cinematic_start()

```
void cinematic_start (
            SDL_Renderer * pRenderer,
            int * next,
            SDL_Rect * P1,
            SDL_Rect * P2,
            SDL_Rect * P3,
            SDL_Rect * P4,
            scene_t * scene,
            SDL_Texture ** storyP1,
            SDL_Texture ** storyP2,
            SDL_Texture ** storyP3,
            SDL_Texture ** storyP4,
            int * skipOn,
            SDL_Texture ** Map1,
            SDL_Texture ** Map2,
            SDL_Texture ** Map3,
            SDL_Texture ** Map4,
            SDL_Texture ** Map5,
            SDL_Texture ** Map6,
            int colli[125][250] )
```

fonction gérant l'affichage et les événements de la cinématique de début de jeu

création de la variable event qui récupère les événements

boucle d'attente d'événement

récupération des touchess relâchées

regarde si la touche entrée a été relâchée

récupération de la position de la souris

regarde si la souris est sur le bouton skip

récupération d'un bouton relâché venant de la souris

vérifie que la souris est sur le bouton skip pour réagir au clic

passe l'histoire, réinitialisation de toutes les variables utilisées ici puis passage sur l'écran de chargement vers l'OP

chargement des textures des map

texte de l'histoire séparée en 2 parties (ici 1ère partie)

texte de l'histoire séparée en 2 parties ici (2ème partie)

chargement des textures des textes

chargement de la texture, déclaration des dimensions et des coordonnées du bouton skip

déclaration des dimensions et des coordonnées des textes

affiche la première partie de l'histoire (texte numéro 1, puis 2 après avoir appuyé sur Entrée)

affiche la deuxième partie de l'histoire (même processus)

affiche la troisième partie de l'histoire (même processus)

affiche la quatrième partie de l'histoire (même processus)

fin de l'histoire, réinitialisation de toutes les variables utilisées ici puis passage sur l'écran de chargement vers l'OP

destruction des images illustrant l'histoire

chargement des textures des map

destruction de toutes les textures créées

délai d'affichage

4.3.2.4 DetectEventsInvSac()

```
void DetectEventsInvSac (
             int * useOk,
             int * yonUse,
             int * yonThrow,
             int * MouseClick,
             int * MouseOver,
             scene_t * scene,
             int w,
             int h,
             SDL_Rect destThrowButton )
```

fonction DetectEventsInvSac qui s'occupera de gérer la détection des événements quand nous sommes sur le menu d'inventaire sur le sac

création de la variable event qui récupère les événements

coordonnés des zones ou se situent les zones d'objet dans le menu

boucle d'attente d'événement

récupération des touches enfoncées

regarde quelle touche est enfoncée

récupération des coordonnées de la souris

regarde si la souris est dans les cases d'objet

regarde si la souris est sur le bouton jeter

regarde si la souris est sur le bouton utiliser

regarde si la souris est sur le bouton oui, non ou sur aucun des deux de la boîte de texte pour jeter l'objet

regarde si la souris est sur un personnage pouvant utiliser un objet

récupération si le bouton de la souris est lâché

regarde si le bouton est relâché sur une zone d'objet de l'inventaire

regarde si le bouton est relâché sur le bouton jeter

regarde si le bouton est relâché sur le bouton oui ou non de la boîte de texte pour jeter l'objet

regarde si le bouton est relâché sur le bouton utiliser

4.3.2.5 DetectEventsInvStuff()

```
void DetectEventsInvStuff (
            int * yonThrow,
            int * MouseClick,
            int * MouseOver,
            scene_t * scene,
            int w,
            int h,
            SDL_Rect destThrowButton,
            int * numPage,
            int * numPagePerso,
            int * slot,
            int * eOuStuff )
```

fonction DetectEventsInvStuff qui s'occupera de gérer la détection des événements quand nous sommes sur le menu d'inventaire pour les équipements/armes

création de la variable event qui récupère les événements

coordonnés des zones ou se situent les zones d'objet dans le menu

coordonnés du personnage affiché

boucle d'attente d'événement

récupération des touches relâchées

regarde quelle touche est relâchée

réinitialise toutes les variables utiliser en même temps de changer de scene

récupération des coordonnées de la souris

regarde si la souris est dans les cases des équipements/armes

regarde si la souris est dans un slot d'équipement

slot 1 et 2

slot 3 et 4

slot 5 et 6

regarde si la souris est sur le bouton équiper ou déséquiper

bouton équiper

bouton déséquiper

si la souris n'est sur aucun bouton

regarde si la souris est sur le bouton jeter

regarde si la souris est sur le bouton flèche droite ou gauche dans l'inventaire

flèche gauche

flèche droite

regarde si la souris est sur le bouton flèche droite ou gauche sur les personnages

flèche gauche

flèche droite

regarde si la souris est sur le bouton oui, non ou sur aucun des deux de la boîte de texte pour jeter l'objet

récupération si le bouton de la souris est lâché

regarde si le bouton est relâché sur une zone d'objet de l'inventaire

regarde si le bouton est relâché sur le bouton équiper ou déséquiper

regarde si le bouton est relâché sur un slot d'équipements/armes

regarde si le bouton est relâché sur le bouton jeter

regarde si le bouton est relâché sur le bouton oui ou non de la boîte de texte pour jeter l'objet

4.3.2.6 DetectEventsOp()

```
void DetectEventsOp (
             int * MouseOver,
             int * KeyIsPressed,
             int * Action,
             scene_t * scene,
             int w,
             int h,
             int * choiceShop,
             int * vendeur,
             int * numperso )
```

fonction DetectEventsOp qui s'occupera de gérer la détection des événements quand nous sommes sur l'openWorld

création de la variable event qui récupère les événements

boucle d'attente d'événement

récupération des touchess enfoncées

regarde quelle touche est enfoncée

change la variable KeyIsPressed si la touche est maintenue

récupération des coordonnées de la souris

regarde si la souris est dans les cases d'objet

récupération des touchess relâchées et changement de KeyIsPressed

regarde quelle touche a été relâchée et change la variable KeyIsPressed en fonction de la touche

touche de déplacement z, q, s, d

touche pour changer de personnage

touche d'interaction

touche pour faire pause

touche d'ouverture d'inventaire (sac)

touche d'ouverture d'inventaire (équipement et armes)

touche d'ouverture des stats

touche flèche du haut pour les choix du vendeur

touche flèche du bas pour les choix du vendeur

touche entrée pour valider les choix du vendeur

détection clic souris relaché

icône d'inventaire (sac d'objet)

icône des stats

icône d'inventaire (équipement et armes)

4.3.2.7 DetectEventsStats()

```
void DetectEventsStats (
            int * numPage,
            int * MouseClick,
            int * MouseOver,
            scene_t * scene,
            int w,
            int h,
            SDL_Rect * destStatAction,
            team_t * team,
            float Stats[3][8] )
```

fonction DetectEventsStats qui s'occupera de gérer la détection des événements quand nous sommes sur le menu des stats des personnages

création de la variable event qui récupère les événements

boucle d'attente d'événement

récupération des touches enfoncées

regarde quelle touche est enfoncée

récupération des coordonnées de la souris

regarde si la souris est sur les flèches pour changer de perso sur le bouton valider

regarde si la souris est sur un bouton +

regarde si la souris est sur un bouton -

regarde si la souris est dans la case + ou - d'une stat

récupération si le bouton de la souris est lâché

regarde si le bouton est relâché sur une zone d'objet de l'inventaire

4.3.2.8 EventSell()

```
void EventSell (
            int * MouseOver,
            int * MouseClick,
            int * Action,
            SDL_Renderer * pRenderer,
            inventaire_t * inv,
            int w,
            int h,
            scene_t * scene,
            int * numPage )
```

déclaration du nombre de pages et des variables pour la taille du texte en largeur et hauteur

création de la variable event qui récupère les événements

boucle d'attente d'événement

regarde la position de la souris

bouton retour (55) et vendre (54)

regarde si la souris est dans les cases d'équipements/armes

regarde si la souris est sur la flèche gauche ou droite

retour aux choix du vendeur si le bouton retour est actionné

bouton vendre actionné seulement si un équipement ou une arme est sélectionné

MouseClick prend la valeur de l'équipement ou de l'arme sélectionné dans l'inventaire

change de page selon la flèche sur laquelle on est

vérification du numéro de page pour éviter de dépasser le nombre de pages ou d'être en-dessous de 0

déclaration des textures

déclaration des couleurs

déclaration rect

déclaration char pour les prix ainsi que la bourse actuelle

chargement de la texture du texte bourse avec l'argent possédé

change l'affichage du bouton retour selon si on est dessus ou non

change l'affichage du bouton vendre selon si on est dessus ou non, si un équipement ou une arme est sélectionné ou non

affichage des bouton acheter et retour, de la zone d'item et de la bourse actuelle en haut à gauche

chargement des textures des noms, descriptions et des prix des équipements/armes dans l'inventaire

regarde si on est sur des équipements

regarde si on est sur des armes

sinon si l'inventaire a atteint sa limite mettre à NULL

affichage des équipements/armes et de la zone sur laquelle la souris a cliqué ou est positionnée

affiche seulement s'il y a un équipements/armes à cet emplacement

affichage de la zone autour des noms selon s'il est sélectionné ou non

mise à jour des coordonnées et dimensions des noms des équipements/armes

affichage des noms des équipements/armes

affichage de l'item sélectionné

affichage de l'image des équipements/armes ainsi que sa description

affichage flèche de gauche dans l'inventaire puis libère la mémoire

calcul des coordonnées/dimensions et affichage de la flèche droite dans l'inventaire puis libère la mémoire

libération de toutes les textures

4.3.2.9 EventShop()

```
void EventShop (
            int * MouseOver,
            int * Action,
            SDL_Renderer * pRenderer,
            int biome,
            equipement_t allStuff[ALL_EQUIPEMENTS][3],
            inventaire_t * inv,
            int w,
            int h,
            scene_t * scene,
            objet_t allPot[ALL_OBJETS],
            int * choiceBiy,
            arme_t armes[ALL_ARMES][3] )
```

fonction EventsInvShop qui gère le menu d'achat dans le magasin pour acheter équipement, armes et objets

création de la variable event qui récupère les événements

boucle d'attente d'événement

regarde la position de la souris

bouton retour (30) et acheter (29)

sur les items de la première étagère

sur les items de la deuxième étagère

sinon sur aucun bouton ou interaction

récupération d'un bouton relâché venant de la souris

bouton retour actionné on revient au choix du vendeur

toutes les réactions si le bouton acheter est actionné ou si un item est sélectionné

biome 1

item sélectionné

si le bouton acheter est cliqué et un objet est sélectionné

on vérifie que l'achat soit possible à chaque fois

cas d'un équipement

cas d'un objet

biome 4

item sélectionné

si le bouton acheter est cliqué et un objet est sélectionné

on vérifie que l'achat soit possible à chaque fois

cas d'un équipement

cas objet

biome 2,3 et 5

item sélectionné

si le bouton acheter est cliqué et un objet est sélectionné

on vérifie que l'achat soit possible à chaque fois

cas d'un équipement

biome 5

autre biome

cas objet

objets différents dans chaque magasin selon le biome

biome 2

biome 3

biome 5

déclaration texture

déclaration rect

déclaration couleur

déclaration char pour les prix ainsi que la bourse actuelle

change l'affichage du bouton retour selon si on est dessus ou non

change l'affichage du bouton acheter selon si on est dessus ou non, si on a l'argent nécessaire pour l'acheter et si on peut le stocker

affichage des boutons acheter et retour, de la zone d'item et de la bourse actuelle en haut à gauche

calcul des coordonnées et affichage des items ainsi que leur prix juste en-dessous

affiche le prix en blanc par défaut, et l'affiche en jaune si l'item est sélectionné

affiche le prix en blanc par défaut, et l'affiche en jaune si l'item est sélectionné

affiche le prix en blanc par défaut, et l'affiche en jaune si l'item est sélectionné

affiche le prix en blanc par défaut, et l'affiche en jaune si l'item est sélectionné

affiche le prix en blanc par défaut, et l'affiche en jaune si l'item est sélectionné

affiche le prix en blanc par défaut, et l'affiche en jaune si l'item est sélectionné

affiche le prix en blanc par défaut, et l'affiche en jaune si l'item est sélectionné

affiche le prix en blanc par défaut, et l'affiche en jaune si l'item est sélectionné

affiche le prix en blanc par défaut, et l'affiche en jaune si l'item est sélectionné

affiche le prix en blanc par défaut, et l'affiche en jaune si l'item est sélectionné

affiche le prix en blanc par défaut, et l'affiche en jaune si l'item est sélectionné

affiche le prix en blanc par défaut, et l'affiche en jaune si l'item est sélectionné

affiche le prix en blanc par défaut, et l'affiche en jaune si l'item est sélectionné

affiche le prix en blanc par défaut, et l'affiche en jaune si l'item est sélectionné

affiche le prix en blanc par défaut, et l'affiche en jaune si l'item est sélectionné

affiche le prix en blanc par défaut, et l'affiche en jaune si l'item est sélectionné

affiche le prix en blanc par défaut, et l'affiche en jaune si l'item est sélectionné

affiche le prix en blanc par défaut, et l'affiche en jaune si l'item est sélectionné

affiche le prix en blanc par défaut, et l'affiche en jaune si l'item est sélectionné

affiche le prix en blanc par défaut, et l'affiche en jaune si l'item est sélectionné

destruction des textures

délai d'affichage

4.3.2.10 ReactEventsInvSac()

```
void ReactEventsInvSac (
            int * useOk,
            int * yonUse,
            int * yonThrow,
            int MouseClick,
            int KeyIsPressed,
            int MouseOver,
            SDL_Renderer * pRenderer,
            int w,
            int h,
            inventaire_t * inv,
            SDL_Rect * destThrowButton,
            perso_t ** team )
```

fonction ReactEventsInvSac qui s'occupe de réagir en fonction des événements quand nous sommes sur le menu d'inventaire sur le sac

déclaration de i servant aux boucles for et textwidth, textheight pour les dimensions des textes

chaîne de caractère utilisée pour l'affichage des persos avec leur nb de pv et pm

chargement des textures nécessaires au menu

chargement des textures nécessaires au bouton jeter et sa boîte

chargement des textures nécessaires au bouton utiliser et sa boîte

déclaration des rect correspondant aux coordonnées et dimensions d'affichage du menu

déclaration des rect correspondant aux coordonnées et dimensions d'affichage des boîtes utiliser et jeter ainsi que des personnages pour la boîte d'utilisation

déclaration et initialisation des textures, des noms et descriptions des objets ainsi que la couleur des textes

déclaration et initialisation des textures des textes des personnages dans la boîte d'utilisation d'objet

déclaration de la variable pour les coordonnées et dimensions des noms d'objet

coordonnées et dimensions de la description des objets

coordonnées et destination du bouton jeter

coordonnées et destination du bouton utiliser

affichage du menu

affichage des objets présent ou non ainsi que la zone sur laquelle la souris a cliqué ou est positionné

mise à jour des coordonnées et dimensions des noms d'objet

affichage des noms d'objet (affiche ¨. . .¨ s'il n'y a pas d'objet)

test s'il y a un objet à cet emplacement du sac, si oui affiche l'image, la description et les boutons jeter et utiliser, sinon affiche le texte ¨aucun objet¨

cas où l'objet sélectionné existe dans cet endroit de l'inventaire

affichage de l'image de l'objet ainsi que sa description

regarde yonThrow pour gérer l'affichage du bouton jeter ainsi que sa boîte d'interaction pour jeter un objet

regarde yonUse pour gérer l'affichage du bouton utiliser ainsi que sa boîte d'interaction pour sélectionner un personnage met useOk à 0 si le personnage peut utiliser cet objet, met la variable à 1 sinon

regarde si un personnage a été sélectionné

cas où l'objet sélectionné n'existe pas dans cet endroit de l'inventaire

destruction des textures utilisées pour le menu

destruction des textures utilisées pour le bouton jeter et sa boîte

destruction des textures utilisées pour le bouton utiliser et sa boîte

destruction des textures des noms et descriptions des objets

destruction des textures des noms, pv et pm des personnages

4.3.2.11 ReactEventsInvStuff()

```
void ReactEventsInvStuff (
            int * yonThrow,
            int MouseClick,
            int MouseOver,
            SDL_Renderer * pRenderer,
            int w,
            int h,
            inventaire_t * inv,
            SDL_Rect * destThrowButton,
            team_t * team,
            int numPage,
            int numPagePerso,
            int slot,
            int * eOuStuff )
```

fonction ReactEventsInvStuff qui s'occupe de réagir en fonction des événements quand nous sommes sur le menu d'inventaire pour les équipements/armes

déclaration de i servant aux boucles for et textwidth, textheight pour les dimensions des textes

calcule le nombre de pages nécessaire selon le nombre d'équipements et d'armes

chargement des textures nécessaires au menu

chargement des textures nécessaires au bouton jeter et sa boîte

chargement des textures nécessaires à la boîte ou se situent le personnage et ses slots d'équipements/armes

déclaration des rect correspondant aux coordonnées et dimensions d'affichage du menu

déclaration des rect correspondant aux coordonnées et dimensions d'affichage du perso et des slots d'équiepements/armes

déclaration des rect correspondant aux coordonnées et dimensions d'affichage des boîtes utiliser et jeter ainsi que des personnages pour la boîte d'utilisation

déclaration de la texture et des coordonnés + dimensions des flèches pour changer de page

déclaration de la texture et des coordonnés + dimensions pour les boutons équiper et déséquiper

déclaration et initialisation des textures des noms et descriptions des équipements/armes ainsi que la couleur des texte jaune si équiper, blanc sinon

regarde si la sourie est sur des équipements

mettre à NULL si l'inventaire a atteint sa limite

déclaration de la variable pour les coordonnées et dimensions des noms d'objet

coordonnées et dimensions de la description des objets

coordonnées et destination du bouton jeter

coordonnées et destination des bouton flèches gauche/droite

affichage du menu

calcul des coordonnées/dimensions de la boîte contenant le personnage et affichage

affichage du personnage selon la page

page 1 Golem

page 2 et 3 Toxo/guerrier et xero/mage

destruction de la texture du golem de face

affichage des slots d'équipements

slot 1

afficher un équiement dans le slot s'il est équipé

slot 2

slot 3

slot 4

affichage des slots d'armes

slot 1

slot 2

affichage des équipements/armes et de la zone sur laquelle la souris a cliqué ou est positionnée

affiche seulement s'il y a un équipements/armes à cet emplacement

affichage de la zone autour des noms selon s'il est sélectionné ou non

mise à jour des coordonnées et dimensions des noms des équipements/armes

affichage des noms des équipements/armes

test s'il y a un équipement ou une arme à cet emplacement d'inventaire, si oui affiche l'image, la description et le bouton jeter, et les flèches pour changer de page si possible

affichage de l'image des équipements/armes ainsi que sa description

regarde yonThrow pour gérer l'affichage du bouton jeter ainsi que sa boîte d'interaction pour jeter un objet

cas où l'on a cliqué sur oui

affichage de la flèche de gauche dans l'inventaire puis libère la mémoire

calcul des coordonnées/dimensions et affichage de la flèche droite dans l'inventaire puis libère la mémoire

calcul des coordonnées/dimensions et affichage de la flèche gauche des persos puis libère la mémoire

calcul des coordonnées/dimensions et affichage de la flèche droite des perso puis libère la mémoire

équipe l'équipement ou l'arme séléctionné dans l'inventaire au personnage correspondant au numéro de page des personnages si le bouton équiper est appuyé

chargement->affichage puis libération de la mémoire pour le bouton équiper

déséquipe l'équipement ou l'arme séléctionné dans les slots du personnage si le bouton déséquiper est appuyé

chargement->affichage puis libération de la mémoire pour le bouton déséquiper

cas équipement

cas armes

destruction des textures utilisées pour le menu

destruction des textures utilisées pour le bouton jeter et sa boîte

destruction des textures utilisées pour les slots des équipements/armes

destruction des textures des noms et descriptions des objets

4.3.2.12 ReactEventsOp()

```
void ReactEventsOp (
            int * avancement_quete,
            int MouseOver,
            int KeyIsPressed,
            int * Action,
            SDL_Renderer * pRenderer,
            SDL_Texture * Perso,
            SDL_Rect marcheHori[3][8],
            SDL_Rect * dest,
            int * i,
            int w,
            int h,
            team_t team,
            int biome,
            int choiceShop,
            int * vendeur,
            int colli[125][250],
            SDL_Texture * vendeurDialoTexture[5],
            int * map,
            SDL_Rect * savedest,
            int numperso )
```

déclaration des fonctions ReactEvents pour les sets d'événement possibles selon la scène

déclaration des fonctions ReactEvents pour les sets d'événement possibles selon la scène déclaration des textes pour les pnj

déclaration de la texture de chaque icône

déclaration de la texture des PV et PM de chaque perso

coordonnées et dimensions des PV et PM de chaque persos

coordonnées et dimensions de chaque icône

déclaration des dimensions et des coordonnées de chaque dialogues

déclaration des dimensions et des coordonnées du perso centré

déclaration des dimensions et des coordonnées de chaque dialogue du vendeur

déclaration de la couleur du texte

bloque le personnage si collision est détecté lors du déplacement vers la gauche

bloque le personnage si collision est détecté lors du déplacement vers la droite

bloque le personnage si collision est détecté lors du déplacement vers le haut

bloque le personnage si collision est détecté lors du déplacement vers le bas

cas où le personnage s'arrête vers la droite

cas où le personnage s'arrête vers la gauche

cas où le personnage s'arrête vers le haut

cas où le personnage s'arrête vers le bas

réinitialisation du i pour éviter qu'il dépasse la valeur max d'un int

teste si le personnage est en face d'un pnj vendeur pour faire une interaction une fois la touche f appuyée

teste si le personnage est en face d'un pnj pour faire une interaction une fois la touche f appuyée

teste si le personnage est en face d'un pnj pour faire une interaction une fois la touche f appuyée

test s'il est possible pour le personnage de changer de zone

réinitialise l'action s'il n'y a aucune interaction avec des pnj

chargement des textures selon si la souris est sur l'icône ou non

chargement, affichage et destruction des têtes des persos ainsi que leur nombre PV et de PM

cas où 1 perso est dans l'équipe

chargement de la barre de PV selon le pourcentage de pv du perso

chargement de la barre de Mana selon le pourcentage de Mana du perso, la barre est grisée si ses PM max sont à 0

cas où 2 persos sont dans l'équipe

chargement des têtes des 2 personnages

affichage des têtes des 2 personnage

chargement de la barre de PV selon le pourcentage de pv du perso 1 puis destruction de leur texture

chargement de la barre de PV selon le pourcentage de pv du perso 2

chargement de la barre de Mana selon le pourcentage de Mana du perso 1, la barre est grisée si ses PM max sont à 0

chargement de la barre de Mana selon le pourcentage de Mana du perso 2, la barre est grisée si ses PM max sont à 0

destruction de toutes les textures

cas où 3 persos sont dans l'équipe

chargement des têtes des 3 personnages

affichage des têtes des 3 personnages

chargement de la barre de PV selon le pourcentage de pv du perso 1 puis destruction de leur texture

chargement de la barre de PV selon le pourcentage de pv du perso 2 puis destruction de leur texture

chargement de la barre de PV selon le pourcentage de pv du perso 3

chargement de la barre de Mana selon le pourcentage de Mana du perso 1, la barre est grisée si ses PM max sont à 0

chargement de la barre de Mana selon le pourcentage de Mana du perso 2, la barre est grisée si ses PM max sont à 0

chargement de la barre de Mana selon le pourcentage de Mana du perso 2, la barre est grisée si ses PM max sont à 0

destruction de toutes les textures

affichage des icônes

destruction des textures des icônes

délai d'affichage

4.3.2.13 ReactEventsStats()

```
void ReactEventsStats (
            int numPage,
            int MouseClick,
            int KeyIsPressed,
            int MouseOver,
            SDL_Renderer * pRenderer,
            int w,
            int h,
            team_t * team,
            float Stats[3][8],
            SDL_Rect * destStatAction )
```

fonction ReactEventsInvSac qui s'occupe de réagir en fonction des événements quand nous sommes sur le menu d'inventaire sur le sac

déclaration de i servant aux boucles for et textwidth, textheight pour les dimensions des textes

chaîne de caractère utilisée pour l'affichage des persos avec leur stats

stats stockées temporairement pour vérifier certaines conditions pour pouvoir utiliser le bouton + et -

chargement des textures nécessaires au menu

déclaration des rect correspondant au coordonnées et dimensions d'affichage du menu

déclaration et initialisation des textures des noms et descriptions des objets ainsi que la couleur des textes

déclaration et initialisation des textures des textes des personnages dans la boîte d'utilisation d'objet

déclaration de la variable pour les coordonnées et dimensions des noms d'objet

affichage du menu

création de la texture du nom du perso avec son level

calcul des coordonnées et dimensions du nom du perso avec son lvl

affichage du nom du perso avec son lvl puis libère la mémoire

calcul des coordonnées, dimensions des bouton + et -

chargement de la texture puis affichage du bouton + en on ou off en fonction de si l'on peut utiliser les bouton ou non pour éviter de dépassez le nombre de pt de compétence à utiliser, et libère la mémoire

chargement de la texture puis affichage du bouton - en on ou off en fonction de si l'on peut utiliser les bouton ou non pour éviter de diminuer en-dessous des stats de base, et libère la mémoire

récupère les dernières coordonnées des boutons + et - pour la détection d'events

mise à jour des coordonnées et dimensions des stats

affichage des stats

calcul des coordonnées, dimensions des points de compétence du personnage sur lequel on est

affichage des points de compétence du personnage

calcul des coordonnées, dimensions, charge les textures puis affichage du bouton valider pour confirmer les points de compétence attribués

calcul des coordonnées, dimensions de la flèche gauche pour changer de perso

affiche les flèches pour changer de perso et assombrit le bouton si la souris est dessus
n'affiche pas la flèche de gauche si nous sommes sur le premier perso et n'affiche pas la flèche de droite si nous sommes sur le dernier perso puis libère la mémoire

calcul des coordonnées, dimensions de la flèche droite pour changer de perso puis libère la mémoire

copie des coordonnées des derniers boutons pour les détections d'events

destruction des textures utilisées pour le menu

destruction des textures des noms et descriptions des objets

destruction de toutes les textures des boutons

4.3.2.14 transitionFondu()

```
void transitionFondu (
             SDL_Renderer * renderer,
             int largeur_ecran,
             int hauteur_ecran,
             int duree )
```

déclaration des fonctions DetectEvents pour les sets d'événement possibles selon la scène

déclaration des fonctions DetectEvents pour les sets d'événement possibles selon la scène Nombre d'images et délai d'affichage

Boucle pour effectuer le fondu

Calculer l'alpha en fonction du progrès de la transition

Affichage du fondu

Mettre à jour l'affichage

Délai d'affichage

4.4 EventOp.h

Go to the documentation of this file.

```
00001
00009 #ifndef EVENTOP_H
00010 #define EVENTOP_H
00011 #include <SDL2/SDL.h>
00012 #include <SDL2/SDL_ttf.h>
00013 #include <SDL2/SDL_image.h>
00014 #include <SDL2/SDL_mixer.h>
00015 #include "type.h"
00016 #include "texture.h"
00017 #include "../lib/fmap.h"
00018 #include "../lib/fpnj.h"
00019 #include "../lib/fmobs.h"
00020 #include "../lib/fbibliotheque.h"
00021
00025 void transitionFondu(SDL_Renderer* renderer, int largeur_ecran, int hauteur_ecran, int duree);
00026 void chargement(scene_t * scene, SDL_Renderer* pRenderer,scene_t futurScene);
00027 void cinematic_start(SDL_Renderer* pRenderer,int * next,SDL_Rect * P1,SDL_Rect * P2,SDL_Rect *
      P3,SDL_Rect * P4, scene_t * scene,SDL_Texture ** storyP1,SDL_Texture ** storyP2,SDL_Texture **
      storyP3,SDL_Texture ** storyP4,int *skipOn,SDL_Texture ** Map1,SDL_Texture ** Map2,SDL_Texture **
      Map3,SDL_Texture ** Map4,SDL_Texture ** Map5,SDL_Texture ** Map6,int colli[125][250]);
00028 void chargement_barre_pv(int ,SDL_Texture** , SDL_Texture** ,SDL_Renderer*);
00029 void DetectEventsOp(int*, int* , int* ,scene_t*,int,int,int*,int*,int*);
00030 void DetectEventsInvSac(int *useOk, int *yonUse,int *yonThrow,int *MouseClick,int *MouseOver, scene_t*
      scene,int w,int h,SDL_Rect destThrowButton);
00031 void DetectEventsStats(int *numPage,int *MouseClick,int *MouseOver, scene_t* scene,int w,int
      h,SDL_Rect* destStatAction,team_t * team,float Stats[3][8]);
00032 void DetectEventsInvStuff(int *yonThrow,int* MouseClick,int *MouseOver, scene_t* scene,int w,int
      h,SDL_Rect destThrowButton,int *numPage,int* numPagePerso,int* slot,int*);
00033
00037 void ReactEventsOp(int *,int,int,int*,SDL_Renderer* ,SDL_Texture*,SDL_Rect marcheHori[3][8],SDL_Rect*
      ,int*,int,int,team_t,int biome,int,int*,int[125][250],SDL_Texture * vendeurDialoTexture[5],int *
      map,SDL_Rect* savedest,int numperso);
00038
00039 void ReactEventsInvSac(int *useOk, int *yonUse,int *yonThrow,int MouseClick, int KeyIsPressed,int
      MouseOver,SDL_Renderer* pRenderer,int w,int h,inventaire_t*,SDL_Rect *destThrowButton, perso_t **
      team);
00040
00041 void ReactEventsStats(int numPage,int MouseClick, int KeyIsPressed,int MouseOver,SDL_Renderer*
      pRenderer,int w,int h,team_t * team,float Stats[3][8],SDL_Rect * destStatAction);
00042
00043 void ReactEventsInvStuff(int *,int ,int ,SDL_Renderer* ,int ,int ,inventaire_t* ,SDL_Rect *,team_t *
      ,int ,int,int,int*);
00044
00045 void EventShop(int *MouseOver,int *Action,SDL_Renderer* pRenderer,int biome,equipement_t
      allStuff[ALL_EQUIPEMENTS][3],inventaire_t* inv,int w,int h,scene_t* scene,objet_t
      allPot[ALL_OBJETS],int*,arme_t armes[ALL_ARMES][3]);
00046 void EventSell(int *MouseOver,int *MouseClick,int *Action,SDL_Renderer* pRenderer,inventaire_t*
      inv,int w,int h,scene_t* scene,int *numPage);
00047 int achat_possible(int choiceBiy,int biome,inventaire_t * inv,objet_t allPot[ALL_OBJETS],equipement_t
      allStuff[ALL_EQUIPEMENTS][3],arme_t armes[ALL_ARMES][3]);
00048 #endif
00049
```

4.5 lib/fbibliotheque.h File Reference

programme gérant les fonctions pour la Map de la bibliotheque

```
#include <SDL2/SDL.h>
#include <SDL2/SDL_image.h>
#include <SDL2/SDL_ttf.h>
#include <SDL2/SDL_mixer.h>
#include <stdio.h>
#include ¨../lib/fmap.h¨
```

Macros

- #define **WINDOW_WIDTH** 1360
- #define **WINDOW_HEIGHT** 780

Functions

- int charger_map_biblio (char ∗CARTE_LIGNE, char ∗CARTE_TXT, char ∗nom_carte, int tab[125][250])
- int charger_map_pnj_biblio (char ∗PNJ_LIGNE, char ∗PNJ_TXT, char ∗MAP, char ∗nom_carte, int collisions[125][250])

4.5.1 Detailed Description

programme gérant les fonctions pour la Map de la bibliotheque

Author

François Lépine.

Version

1.9.5

Date

16 Avril 2024.

4.5.2 Function Documentation

4.5.2.1 charger_map_biblio()

```
int charger_map_biblio (
            char * CARTE_LIGNE,
            char * CARTE_TXT,
            char * nom_carte,
            int tab[125][250] )
```

Ouverture du fichier

Ouvre le fichier tuiles_plaine qui contient les tuiles

Gestion de l'erreur

Variables pour stocker les nombres

Définir la portion de l'image à afficher

Largeur de la portion à afficher

Hauteur de la portion à afficher

Définir le rectangle de destination

Utiliser la même largeur que la source

Utiliser la même hauteur que la source

Récupération de la largeur det de la hauteur

Création de la variable qui va créer le .png contenant l'image de la map

Parcours du fichier contenant la carte en numéro de tuile

Décalage de l'affichage (x et y) sur la map pour pouvoir afficher toute la map

Affichage de la tuile

On va pour chaque tuile enregistrer dans un tableau

Fermeture du fichier

Sauvegarde de la carte en png

4.5.2.2 charger_map_pnj_biblio()

```
int charger_map_pnj_biblio (
            char * PNJ_LIGNE,
            char * PNJ_TXT,
            char * MAP,
            char * nom_carte,
            int collisions[125][250] )
```

Ouverture du fichier

Ouvre le fichier pnj_ligne qui contient les pnj

Gestion de l'erreur

Ouvre le fichier de la carte du jeu pour y ajouter les pnj

Gestion de l'erreur

Variables pour stocker les nombres

Définir la portion de l'image à afficher

Largeur de la portion à afficher

Hauteur de la portion à afficher

Définir le rectangle de destination

Utiliser la même largeur que la source

Utiliser la même hauteur que la source

Récupération de la largeur det de la hauteur

Parcours du fichier contenant la carte en numéro de tuile

Affichage de la tuile

Fermeture du fichier

Sauvegarde de la carte en png

4.6 fbibliotheque.h

Go to the documentation of this file.

```
00001
00009 #include <SDL2/SDL.h>
00010 #include <SDL2/SDL_image.h>
00011 #include <SDL2/SDL_ttf.h>
00012 #include <SDL2/SDL_mixer.h>
00013 #include <stdio.h>
00014 #include "../lib/fmap.h"
00015
00016 #define WINDOW_WIDTH 1360
00017 #define WINDOW_HEIGHT 780
00018
00019 int charger_map_biblio(char * CARTE_LIGNE,char * CARTE_TXT,char * nom_carte, int tab[125][250]);
00020 int charger_map_pnj_biblio(char * PNJ_LIGNE,char * PNJ_TXT,char * MAP,char * nom_carte,int
      collisions[125][250]);
```

4.7 lib/fmap.h File Reference

programme gérant les fonctions pour la Map

```
#include <SDL2/SDL.h>
#include <SDL2/SDL_image.h>
#include <SDL2/SDL_ttf.h>
#include <SDL2/SDL_mixer.h>
#include <stdio.h>
```

Macros

- #define **WINDOW_WIDTH** 1360
- #define **WINDOW_HEIGHT** 780

Functions

- SDL_Surface ∗ **loadTileset** (const char ∗filename)
- int **saveSurfaceAsPNG** (SDL_Surface ∗surface, const char ∗filename)
- int charger_map (char ∗CARTE_LIGNE, char ∗CARTE_TXT, char ∗nom_carte, int tab[125][250])
- int afficher_map (int coordo_x, int coordo_y, SDL_Renderer ∗renderer, SDL_Texture ∗)

4.7.1 Detailed Description

programme gérant les fonctions pour la Map

Author

François Lépine.

Version

1.9.5

Date

16 Avril 2024.

4.7.2 Function Documentation

4.7.2.1 afficher_map()

```
int afficher_map (
            int coordo_x,
            int coordo_y,
            SDL_Renderer * renderer,
            SDL_Texture * texture_carte )
```

Définir la portion de l'image à afficher

Largeur de la portion à afficher

Hauteur de la portion à afficher

Définir le rectangle de destination

Utiliser la même largeur que la source

Utiliser la même hauteur que la source

4.7.2.2 charger_map()

```
int charger_map (
            char * CARTE_LIGNE,
            char * CARTE_TXT,
            char * nom_carte,
            int tab[125][250] )
```

Ouverture du fichier

Ouvre le fichier tuiles_plaine qui contient les tuiles

Gestion de l'erreur

Variables pour stocker les nombres

Définir la portion de l'image à afficher

Largeur de la portion à afficher

Hauteur de la portion à afficher

Définir le rectangle de destination

Utiliser la même largeur que la source

Utiliser la même hauteur que la source

Récupération de la largeur det de la hauteur

Création de la variable qui va créer le .png contenant l'image de la map

Parcours du fichier contenant la carte en numéro de tuile

Décalage de l'affichage (x et y) sur la map pour pouvoir afficher toute la map

Affichage de la tuile

On va pour chaque tuile enregistrer dans un tableau

Fermeture du fichier

Sauvegarde de la carte en png

4.8 fmap.h

Go to the documentation of this file.

```
00001
00009 #include <SDL2/SDL.h>
00010 #include <SDL2/SDL_image.h>
00011 #include <SDL2/SDL_ttf.h>
00012 #include <SDL2/SDL_mixer.h>
00013 #include <stdio.h>
00014
00015 #define WINDOW_WIDTH 1360
00016 #define WINDOW_HEIGHT 780
00017
00018 SDL_Surface* loadTileset(const char* filename);
00019 int saveSurfaceAsPNG(SDL_Surface* surface, const char* filename);
00020 int charger_map(char * CARTE_LIGNE,char * CARTE_TXT,char * nom_carte, int tab[125][250]);
00021 int afficher_map(int coordo_x,int coordo_y,SDL_Renderer* renderer,SDL_Texture*);
```

4.9 fmobs.h

```
00001
00009 #include <SDL2/SDL.h>
00010 #include <SDL2/SDL_image.h>
00011 #include <SDL2/SDL_ttf.h>
00012 #include <SDL2/SDL_mixer.h>
00013 #include <stdio.h>
00014
00015 int copie_fichier_mob_txt(char * fichier_mob_origine,char* fichier_mob_destination);
00016 int charger_map_mobs(char * MOBS_LIGNE,char * MOBS_TXT,char * MOBS_TXT2,char * MAP,int
      collisions[125][250]);
00017 int detruire_mob(int coordo_x_mob_suppr,int coordo_y_mob_suppr,char * MOBS_TXT,int
      collisions[125][250]);
```

4.10 lib/fpnj.h File Reference

programme gérant les fonctions pour les pnj sur la Map

```
#include <SDL2/SDL.h>
#include <SDL2/SDL_image.h>
#include <SDL2/SDL_ttf.h>
#include <SDL2/SDL_mixer.h>
#include <stdio.h>
```

Functions

- int charger_map_pnj (char *PNJ_LIGNE, char *PNJ_TXT, char *MAP, char *nom_carte, int collisions[125][250])

4.10.1 Detailed Description

programme gérant les fonctions pour les pnj sur la Map

Author

François Lépine.

Version

1.9.5

Date

16 Avril 2024.

4.10.2 Function Documentation

4.10.2.1 charger_map_pnj()

```
int charger_map_pnj (
            char * PNJ_LIGNE,
            char * PNJ_TXT,
            char * MAP,
            char * nom_carte,
            int collisions[125][250] )
```

Ouverture du fichier

Ouvre le fichier pnj_ligne qui contient les pnj

Gestion de l'erreur

Ouvre le fichier de la carte du jeu pour y ajouter les pnj

Gestion de l'erreur

Variables pour stocker les nombres

Définir la portion de l'image à afficher

Largeur de la portion à afficher

Hauteur de la portion à afficher

Définir le rectangle de destination

Utiliser la même largeur que la source

Utiliser la même hauteur que la source

Récupération de la largeur det de la hauteur

Parcours du fichier contenant la carte en numéro de tuile

Affichage de la tuile

Fermeture du fichier

Sauvegarde de la carte en png

4.11 fpnj.h

Go to the documentation of this file.

```
00001
00009 #include <SDL2/SDL.h>
00010 #include <SDL2/SDL_image.h>
00011 #include <SDL2/SDL_ttf.h>
00012 #include <SDL2/SDL_mixer.h>
00013 #include <stdio.h>
00014
00015 int charger_map_pnj(char * PNJ_LIGNE,char * PNJ_TXT,char * MAP,char * nom_carte,int
      collisions[125][250]);
```

4.12 lib/init_menu.h File Reference

programme gérant les fonctions pour les menus

```
#include <SDL2/SDL.h>
#include <SDL2/SDL_image.h>
#include <SDL2/SDL_ttf.h>
#include <SDL2/SDL_mixer.h>
#include <stdbool.h>
#include ¨type.h¨
#include ¨texture.h¨
#include ¨fmap.h¨
```

Classes

- struct Image

  Structure pour stocker les informations d'une image.

Macros

- #define **WINDOW_WIDTH** 1360

  Dimensions de la fenêtre.
- #define **WINDOW_HEIGHT** 780
- #define **NOTHING** -1

  Constante de sortie vide.
- #define **PLAY** 1

  Constantes de sortie du menu de l'écran-titre (menu principal)
- #define **SETTINGS** 2
- #define **GAMECREDITS** 3
- #define **BACK** 0

  Constantes de sortie des menus Play, Settings et Credits.
- #define **SAVE1** 10
- #define **SAVE2** 20
- #define **SAVE3** 30
- #define **LOAD** 2
- #define **NEWGAME** 1
- #define **ERASE** 3
- #define **SAVE** 1

  Constantes de sortie du menu de pause.
- #define **SQUIT** 2
- #define **OPTIONS** 3
- #define **RESUME** 4

Functions

- int menuPrincipal (SDL_Renderer ∗renderer)

  Fonctions des menus.
- int menuPlay (SDL_Renderer ∗renderer)

  Menu PLAY.
- int menuSettings (SDL_Renderer ∗renderer)

  Menu SETTINGS.
- int menuCredits (SDL_Renderer ∗renderer)

  Menu CREDITS.
- int menuPause (SDL_Renderer ∗renderer, scene_t ∗scene, SDL_Texture ∗map, SDL_Rect)

  Menu PAUSE.

4.12.1 Detailed Description

programme gérant les fonctions pour les menus

Author

Warrick Bonga.

Version

1.9.5

Date

16 Avril 2024.

4.12.2 Function Documentation

4.12.2.1 menuCredits()

```
int menuCredits (
            SDL_Renderer * renderer )
```

Menu CREDITS.

Chargement de l'image du bouton retour par defaut

Chargement de l'image du bouton retour au survol

Chargement de l'image du bouton retour lors du clic

Effacement de l'écran

Affichage de l'image appropriée

Affichage sur l'écran

Libération des ressources

4.12.2.2 menuPause()

```
int menuPause (
            SDL_Renderer * renderer,
            scene_t * scene,
            SDL_Texture * map,
            SDL_Rect dest )
```

Menu PAUSE.

Chargement de l'image de sauvegarde par defaut

Chargement de l'image du bouton quitter par defaut

Chargement de l'image Options par defaut

Chargement de l'image du bouton retour par défaut

Chargement de l'image de sauvegarde au survol

Chargement de l'image du bouton quitter au survol

Chargement de l'image Options au survol

Chargement de l'image du bouton retour au survol

Chargement de l'image de sauvegarde lors du clic

Chargement de l'image du bouton quitter lors du clic

Chargement de l'image Options lors du clic

Chargement de l'image du bouton retour lors du clic

condition d'arrêt : sélection d'un choix du menu de pause OU appui sur [échap]

Effacement de l'écran

Affichage de l'image appropriée

Affichage sur l'écran

Libération des ressources

4.12.2.3 menuPlay()

```
int menuPlay (
            SDL_Renderer * renderer )
```

Menu PLAY.

Chargement du fond du menu play

Chargement de l'image de la sauvegarde 1 par défaut

Chargement de l'image de la sauvegarde 2 par défaut

Chargement de l'image de la sauvegarde 3 par défaut

Chargement de l'image de la sauvegarde 1 au survol

Chargement de l'image de la sauvegarde 2 au survol

Chargement de l'image de la sauvegarde 3 au survol

Chargement de l'image de la sauvegarde 1 au clic

Chargement de l'image de la sauvegarde 2 au clic

Chargement de l'image de la sauvegarde 3 au clic

Chargement de l'image de l'option lancement par défaut

Chargement de l'image de l'option écrasement par défaut

Chargement de l'image de l'option retour par défaut

Chargement de l'image de l'option lancement au survol

Chargement de l'image de l'option écrasement au survol

Chargement de l'image de l'option retour au survol

Chargement de l'image de l'option lancement au clic

Chargement de l'image de l'option écrasement au clic

Chargement de l'image de l'option retour au clic

condition d'arrêt : chargement d'une sauvegarde (vide ou commencée) OU retour à l'écran-titre

Effacement de l'écran

Affichage de l'image appropriée

Affichage sur l'écran

Libération des ressources

4.12.2.4 menuPrincipal()

```
int menuPrincipal (
            SDL_Renderer * renderer )
```

Fonctions des menus.

Fonctions des menus. Chargement de l'image Play par defaut

Chargement de l'image Settings par defaut

Chargement de l'image Credits par defaut

Chargement de l'image Play au survol

Chargement de l'image Settings au survol

Chargement de l'image Credits au survol

Chargement de l'image Play lors du clic

Chargement de l'image Settings lors du clic

Chargement de l'image Credits lors du clic

condition d'arrêt : choix d'un menu

Effacement de l'écran

Affichage de l'image appropriée

Affichage sur l'écran

Libération des ressources

4.12.2.5 menuSettings()

```
int menuSettings (
            SDL_Renderer * renderer )
```

Menu SETTINGS.

Chargement de l'image du bouton retour par defaut

Chargement de l'image du bouton retour au survol

Chargement de l'image du bouton retour lors du clic

Effacement de l'écran

Affichage de l'image appropriée

Affichage sur l'écran

Libération des ressources

4.13 init_menu.h

Go to the documentation of this file.

```
00001
00009 #include <SDL2/SDL.h>
00010 #include <SDL2/SDL_image.h>
00011 #include <SDL2/SDL_ttf.h>
00012 #include <SDL2/SDL_mixer.h>
00013 #include <stdbool.h>
00014 #include "type.h"
00015 #include "texture.h"
00016 #include "fmap.h"
00017
00019 #define WINDOW_WIDTH 1360
00020 #define WINDOW_HEIGHT 780
00021
00023 #define NOTHING -1
00024
00026 #define PLAY 1
00027 #define SETTINGS 2
00028 #define GAMECREDITS 3
00029
00031 #define BACK 0
00032 #define SAVE1 10
00033 #define SAVE2 20
00034 #define SAVE3 30
00035 #define LOAD 2
00036 #define NEWGAME 1
00037 #define ERASE 3
00038
00040 #define SAVE 1
00041 #define SQUIT 2
00042 #define OPTIONS 3
00043 #define RESUME 4
00044
00046 typedef struct {
00047     SDL_Texture *texture;
00048     SDL_Rect rect;
00049 } Image;
00050
00052 int menuPrincipal(SDL_Renderer *renderer);
00053 int menuPlay(SDL_Renderer *renderer);
00054 int menuSettings(SDL_Renderer *renderer);
00055 int menuCredits(SDL_Renderer *renderer);
00056 int menuPause(SDL_Renderer *renderer, scene_t * scene, SDL_Texture * map,SDL_Rect);
```

4.14 lib/texture.h File Reference

programme gérant les fonctions pour le chargement des textures

```
#include <SDL2/SDL.h>
#include <SDL2/SDL_ttf.h>
#include <SDL2/SDL_image.h>
#include <SDL2/SDL_mixer.h>
```

Functions

- SDL_Texture * loadTexture (const char *, SDL_Renderer *)

  déclaration de la fonction loadTexture qui prend en paramètre le chemin de l'image et le renderer déclaration de la fonction loadTextureFont qui prend en paramètre le chemin de la police d'écriture, le renderer, le texte à afficher, la taille de police et la couleur du texte
- SDL_Texture * loadTextureFont (const char *, SDL_Renderer *, const char *, int, SDL_Color)

  fonction loadTextureFont qui s'occupera de charger le texte avec une surface puis d'utiliser cette surface pour créer une texture la fonction renvoie NULL si il y a une erreur, renvoie la texture créée sinon

4.14.1 Detailed Description

programme gérant les fonctions pour le chargement des textures

Author

Luka Cognard.

Version

1.9.5

Date

16 Avril 2024.

4.14.2 Function Documentation

4.14.2.1 loadTexture()

```
SDL_Texture * loadTexture (
             const char * chemin,
             SDL_Renderer * renderer )
```

déclaration de la fonction loadTexture qui prend en paramètre le chemin de l'image et le renderer déclaration de la fonction loadTextureFont qui prend en paramètre le chemin de la police d'écriture, le renderer, le texte à afficher, la taille de police et la couleur du texte

déclaration de la fonction loadTexture qui prend en paramètre le chemin de l'image et le renderer déclaration de la fonction loadTextureFont qui prend en paramètre le chemin de la police d'écriture, le renderer, le texte à afficher, la taille de police et la couleur du texte crée la surface de l'image, renvoie NULL avec une erreur si ça n'a pas marché

crée la texture de l'image à partir de ça surface puis libère la surface inutiliser renvoie NULL avec une erreur si la texture n'est pas créée

4.14.2.2 loadTextureFont()

```
SDL_Texture * loadTextureFont (
             const char * chemin,
             SDL_Renderer * renderer,
             const char * text,
             int h,
             SDL_Color color )
```

fonction loadTextureFont qui s'occupera de charger le texte avec une surface puis d'utiliser cette surface pour créer une texture la fonction renvoie NULL si il y a une erreur, renvoie la texture créée sinon

charge la police d'écriture via le chemin entré en paramètre renvoie NULL avec une erreur si ça n'a pas marché

crée la surface du texte via le texte entré en paramètre libère la police chargée inutilisée renvoie NULL avec une erreur si ça n'a pas marché

crée la texture du texte à partir de sa surface puis libère la surface inutilisée renvoie NULL avec une erreur si la texture n'est pas créée

4.15 texture.h

Go to the documentation of this file.

```
00001
00009 #ifndef TEXTURE_H
00010 #define TEXTURE_H
00011 #include <SDL2/SDL.h>
00012 #include <SDL2/SDL_ttf.h>
00013 #include <SDL2/SDL_image.h>
00014 #include <SDL2/SDL_mixer.h>
00015
00021 SDL_Texture* loadTexture(const char* , SDL_Renderer* );
00022 SDL_Texture* loadTextureFont(const char* , SDL_Renderer* ,const char*,int,SDL_Color);
00023 #endif
00024
```

4.16 lib/type.h File Reference

programme gérant les fonctions pour les types

```
#include <SDL2/SDL.h>
#include <SDL2/SDL_ttf.h>
#include <SDL2/SDL_image.h>
#include <SDL2/SDL_mixer.h>
#include <stdio.h>
```

Classes

- struct equipement_s

  déclaration du type equipement_t pour donner les stats au stuff
- struct attaque_s
- struct arme_s
- struct perso_s

  déclaration du type perso pour les stats, animation, etc...
- struct pnj_s

  déclaration du type pnj pour leur texture dialogue etc
- struct mob_s
- struct objet_s

  déclaration objet_t qui sera pour les potions et objets utilisables
- struct inventaire_s

  déclaration inventaire_t servant à gérer les objets et équipements possédés par le joueur
- struct team_s

  déclaration team_t pour gérer l'équipe des personnages

Macros

- #define **TAILLE_NOM** 25
- #define **TAILLE_EQUIP** 4
- #define **NB_ARMES** 2
- #define **NB_ATT** 4
- #define **NB_INV_EQUIPEMENTS** 36
- #define **NB_INV_ARMES** 18
- #define **NB_INV_OBJETS** 10
- #define **NB_TEAM_PERSOS** 3
- #define **ALL_OBJETS** 9
- #define **ALL_EQUIPEMENTS** 30
- #define **ALL_ARMES** 28

Typedefs

- typedef struct perso_s **perso_t**

  déclaration du type perso_t pour l'utiliser dans equipement_t et arme_t
- typedef enum scene_s **scene_t**

  déclaration du type scene_t pour différencier les events, l'affichage etc...
- typedef enum typeStuff_s **typeStuff_t**

  déclaration des types de stuff entre multiplicateur et somme
- typedef enum typeArmes_s **typeArmes_t**

  déclaration des types d'arme pour éviter qu'un mage ait une épée par exemple
- typedef enum effet_s **effet_t**
- typedef enum type_att_s **type_att_t**
- typedef struct equipement_s **equipement_t**

  déclaration du type equipement_t pour donner les stats au stuff
- typedef struct attaque_s **attaque_t**
- typedef struct arme_s **arme_t**
- typedef struct pnj_s **pnj_t**

  déclaration du type pnj pour leur texture dialogue etc
- typedef struct mob_s **mob_t**
- typedef struct objet_s **objet_t**

  déclaration objet_t qui sera pour les potions et objets utilisables
- typedef struct inventaire_s **inventaire_t**

  déclaration inventaire_t servant à gérer les objets et équipements possédés par le joueur
- typedef struct team_s **team_t**

  déclaration team_t pour gérer l'équipe des personnages

Enumerations

- enum scene_s {
  MENUP =0 , **PLAYSAVE** , **PARAMETRE** , **CREDITS** ,
  CINEMDEB , **OP** , **PAUSE** , **INVENTORY** ,
  EQUIPEMENT , **STAT** , **FIGHT** , **CINEMEND** ,
  SHOP , **SELL** }

  déclaration du type scene_t pour différencier les events, l'affichage etc...
- enum typeStuff_s { **CASQUE** =0 , **PLASTRON** , **BOTTE** , **TALISMANT** }

  déclaration des types de stuff entre multiplicateur et somme
- enum typeArmes_s { **EPEE** =0 , **ARC** , **BATON** , **CAILLOU** }

  déclaration des types d'arme pour éviter qu'un mage ait une épée par exemple
- enum **effet_s** {
  SOIN =0 , **TAUX_CRIT** , **FOR** , **INT** ,
  DEF , **RES** , **MANA** , **VIT** ,
  ATTAQUE }
- enum **type_att_s** { **PHY** =0 , **MAG** , **STA** , **ERR** }

Functions

- int creer_perso (perso_t *, char *, char *, SDL_Renderer *, float *statPerso)

  déclaration des fonctions liées aux personnages (équiper un équipement/arme ou utiliser un objet, lvl up)
- void **lvl_up** (perso_t *, float *statPerso)

  met a jour le perso quand il a assez d'xp pour lvl up
- void **aff_perso** (perso_t perso)

  affichage d'un perso pour des tests ou autre (effaçable une fois inutile)
- void equiper_stuff (equipement_t *equipement, perso_t *perso)

  fonction qui équipe un équipement d'un personnage
- int **desequiper_stuff** (int iStuff, perso_t *perso)

  fonction qui déséquipe un équipement d'un personnage
- void **utiliser_obj** (inventaire_t *, perso_t *, int)

  fonction qui va utiliser l'objet et le retirer de l'inventaire
- int **join_team** (team_t *, perso_t *)

  déclaration des fonctions liées à l'équipe du joueur
- void **init_reinit__team** (team_t *)
- void **creer_all_objet** (objet_t *, SDL_Renderer *)

  déclaration des fonctions liées aux objets
- attaque_t **creer_attaque** (char nom[TAILLE_NOM], type_att_t type_att, char desc[100], effet_t effet, int qte_effet, int duree_effet, int nb_cible, int precision)

  declaration de la fonction creer_attaque
- void creer_all_sprite_equipement (SDL_Texture *allSprite[ALL_EQUIPEMENTS], SDL_Renderer *ecran)

  déclaration des fonctions liées aux équipements
- void creer_all_equipement (equipement_t allStuff[ALL_EQUIPEMENTS][3], SDL_Texture *allSprite[ALL_EQUIPEMENTS])

  créer le tableau avec tous les équipements en 3 fois pour éviter qu'un personnage soit équipé d'exactement le même équipement
- void init_and_reinit_inv (inventaire_t *inv)

  déclaration des fonction liées à l'inventaire avec le sac d'objets, les équipements et les armes
- int **jeter_obj** (inventaire_t *, int)

  fonction supprime un objet de l'inventaire
- int **ajouter_obj** (inventaire_t *, objet_t *, int)

  fonction ajoute un objet dans l'inventaire
- int ajouter_stuff (equipement_t allStuff[ALL_EQUIPEMENTS][3], inventaire_t *, int iStuff)

  fonction qui ajoute un équipement dans l'inventaire
- int jeter_stuff (inventaire_t *Inv, int iStuff)

  fonction qui supprime un équipement de l'inventaire
- void afficher_arme (arme_t *arme)

  declaration des fonctions en rapport avec les armes et les attaques
- void **creer_all_arme** (arme_t armes[ALL_ARMES][3], SDL_Renderer *ecran)

  fonction qui crée toutes les armes
- void **creer_all_att_arme** (arme_t armes[ALL_ARMES][3])

  fonction qui crée les attaques de chaque arme
- void equiper_arme (arme_t *arme, perso_t *perso)

  fonction qui équipe un équipement d'un personnage
- int **desequiper_arme** (arme_t *arme, perso_t *perso)

  fonction qui déséquipe un équipement d'un personnage
- int ajouter_arme (arme_t allArmes[ALL_ARMES][3], inventaire_t *inv, int indArm)

  fonction qui ajoute une arme dans l'inventaire
- int **jeter_arme** (inventaire_t *Inv, arme_t *arme)

  fonction qui supprime une arme de l'inventaire
- mob_t * creer_mob (char *nom, SDL_Renderer *ecran, int lvl)

  déclaration des fonctions en rapport avec les mobs

4.16.1 Detailed Description

programme gérant les fonctions pour les types

Author

Luka Cognard, Lenny Borry.

Version

1.9.5

Date

16 Avril 2024.

4.16.2 Function Documentation

4.16.2.1 afficher_arme()

```
void afficher_arme (
            arme_t * arme )
```

declaration des fonctions en rapport avec les armes et les attaques

declaration des fonctions en rapport avec les armes et les attaques

4.16.2.2 ajouter_arme()

```
int ajouter_arme (
            arme_t allArmes[ALL_ARMES][3],
            inventaire_t * inv,
            int indArm )
```

fonction qui ajoute une arme dans l'inventaire

regarde si l'inventaire n'est pas plein

regarde combien de fois il possède cet équipement, impossible d'ajouter l'objet s'il l'a 3 fois

ajoute l'équipement avec le numId qui n'est pas présent

renvoie 2 s'il a déja 3 fois l'équipement voulu

renvoie 1 si l'inventaire est plein

4.16.2.3 ajouter_stuff()

```
int ajouter_stuff (
            equipement_t allStuff[ALL_EQUIPEMENTS][3],
            inventaire_t * inv,
            int iStuff )
```

fonction qui ajoute un équipement dans l'inventaire

regarde si l'inventaire n'est pas plein

regarde combien de fois il possède cet équipement, impossible d'ajouter l'objet s'il l'a 3 fois

ajoute l'équipement avec le numId qui n'est pas présent

renvoie 2 s'il a déja 3 fois l'équipement voulu

renvoie 1 si l'inventaire est plein

4.16.2.4 creer_all_equipement()

```
void creer_all_equipement (
            equipement_t allStuff[ALL_EQUIPEMENTS][3],
            SDL_Texture * allSprite[ALL_EQUIPEMENTS] )
```

créer le tableau avec tous les équipements en 3 fois pour éviter qu'un personnage soit équipé d'exactement le même équipement

créer tout les casques, plastrons et bottes par biome

créer les talismans

4.16.2.5 creer_all_sprite_equipement()

```
void creer_all_sprite_equipement (
            SDL_Texture * allSprite[ALL_EQUIPEMENTS],
            SDL_Renderer * ecran )
```

déclaration des fonctions liées aux équipements

déclaration des fonctions liées aux équipements

4.16.2.6 creer_mob()

```
mob_t * creer_mob (
            char * nom,
            SDL_Renderer * ecran,
            int lvl )
```

déclaration des fonctions en rapport avec les mobs

change les stats de base selon la classe du perso

4.16.2.7 creer_perso()

```
int creer_perso (
            perso_t * perso,
            char * nom,
            char * classe,
            SDL_Renderer * ecran,
            float * statPerso )
```

déclaration des fonctions liées aux personnages (équiper un équipement/arme ou utiliser un objet, lvl up)

déclaration des fonctions liées aux personnages (équiper un équipement/arme ou utiliser un objet, lvl up) change les stats de base selon la classe du perso

initialise les tableaux de pointeurs

copie les stats des persos

4.16.2.8 equiper_arme()

```
void equiper_arme (
            arme_t * arme,
            perso_t * perso )
```

fonction qui équipe un équipement d'un personnage

si l'arme est équipée sur un personnage, on la déséquipe

4.16.2.9 equiper_stuff()

```
void equiper_stuff (
            equipement_t * equipement,
            perso_t * perso )
```

fonction qui équipe un équipement d'un personnage

le déséquipe si l'équipement est équipé sur un personnage

le déséquipe de cet objet si un équipement du même type est déjà équipé

4.16.2.10 init_and_reinit_inv()

```
void init_and_reinit_inv (
            inventaire_t * inv )
```

déclaration des fonction liées à l'inventaire avec le sac d'objets, les équipements et les armes

déclaration des fonction liées à l'inventaire avec le sac d'objets, les équipements et les armes

4.16.2.11 jeter_stuff()

```
int jeter_stuff (
            inventaire_t * Inv,
            int iStuff )
```

fonction qui supprime un équipement de l'inventaire

si l'équipement est équipé sur un personnage le déséquipe avant de jeter l'équipement

4.17 type.h

Go to the documentation of this file.

```
00001
00009 #ifndef TYPE_H
00010 #define TYPE_H
00011 #include <SDL2/SDL.h>
00012 #include <SDL2/SDL_ttf.h>
00013 #include <SDL2/SDL_image.h>
00014 #include <SDL2/SDL_mixer.h>
00015 #include <stdio.h>
00016 #define TAILLE_NOM 25
00017 #define TAILLE_EQUIP 4
00018 #define NB_ARMES 2
00019 #define NB_ATT 4
00020 #define NB_INV_EQUIPEMENTS 36
00021 #define NB_INV_ARMES 18
00022 #define NB_INV_OBJETS 10
00023 #define NB_TEAM_PERSOS 3
00024 #define ALL_OBJETS 9
00025 #define ALL_EQUIPEMENTS 30
00026 #define ALL_ARMES 28
00027
00029 typedef struct perso_s perso_t;
00030
00032 typedef enum
      scene_s{MENUP=0,PLAYSAVE,PARAMETRE,CREDITS,CINEMDEB,OP,PAUSE,INVENTORY,EQUIPEMENT,STAT,FIGHT,CINEMEND,SHOP,SELL}scene_t;
00033
00035 typedef enum typeStuff_s{CASQUE=0,PLASTRON,BOTTE,TALISMANT}typeStuff_t;
00036
00038 typedef enum typeArmes_s{EPEE=0,ARC,BATON,CAILLOU}typeArmes_t;
00039
00040 typedef enum effet_s{SOIN=0,TAUX_CRIT,FOR,INT,DEF,RES,MANA,VIT,ATTAQUE}effet_t;
00041
00042 typedef enum type_att_s{PHY=0,MAG,STA,ERR}type_att_t;
00043
00045 typedef struct equipement_s{
00046     char nom[TAILLE_NOM];
00047     char desc[100];
00048     typeStuff_t typeEquipement;
00052     int numId;
00053     perso_t* persoEquiper;
00054     float Pv;
00055     float Mana;
00056     float Def;
00057     float Res;
00058     float For;
00059     float Int;
00060     float Vit;
00061     SDL_Texture * artwork;
00063     int cout;
00064 }equipement_t;
00065
00066 typedef struct attaque_s{
00067     char nom[TAILLE_NOM];
00068     type_att_t type_att;
00069     char desc[100];
00070     effet_t effet;
00071     int qte_effet;
00072     int duree_effet;
00073     int nb_cible;
00074     int precision;
00075
00076 }attaque_t;
00077
00078 typedef struct arme_s{
```

```
00079     char nom[TAILLE_NOM];
00080     char desc[100];
00084     int numId;
00085     perso_t* persoEquiper;
00086     typeArmes_t typeArme;
00087     attaque_t att;
00088     SDL_Texture * artwork;
00090     int cout;
00091 }arme_t;
00092
00094 struct perso_s{
00095     char Nom[TAILLE_NOM];
00096     char Classe[20];
00097     int lvl;
00098     int Xplvl;
00099     int Xp;
00100     int PtComp;
00101     float PvMax;
00102     float Pv;
00103     float ManaMax;
00104     float Mana;
00105     float Def;
00106     float Res;
00107     float For;
00108     float Int;
00109     float Vit;
00110     SDL_Texture* MarcheHori;
00111     SDL_Texture* MarcheFront;
00112     SDL_Texture* MarcheBack;
00113     SDL_Texture* IdleAnim;
00114     SDL_Texture* AttAnim;
00115     equipement_t * stuff[TAILLE_EQUIP];
00116     arme_t * armes[NB_ARMES];
00117     attaque_t attaques[NB_ATT];
00118
00119 };
00120
00122 typedef struct pnj_s{
00123     SDL_Texture* Dialogue[6];
00124     int NbDialogue;
00125 }pnj_t;
00126
00127 typedef struct mob_s{
00128     char nom[TAILLE_NOM];
00129     int lvl;
00130     int Xp_don;
00131     float PvMax;
00132     float Pv;
00133     float ManaMax;
00134     float Mana;
00135     float Def;
00136     float Res;
00137     float For;
00138     float Int;
00139     float Vit;
00140     SDL_Texture* IdleAnim;
00141     attaque_t attaques[NB_ATT];
00142 }mob_t;
00143
00145 typedef struct objet_s{
00146     char nom[13];
00147     char desc[100];
00148     float valPv;
00149     float valMp;
00150     int cout;
00151     SDL_Texture* sprite;
00152 }objet_t;
00153
00155 typedef struct inventaire_s{
00156     int nbObjActu;
00157     objet_t * sac[NB_INV_OBJETS];
00158     int nbEquActu;
00159     equipement_t * equipements[NB_INV_EQUIPEMENTS];
00160     int nbArmActu;
00161     arme_t * armes[NB_INV_ARMES];
00162     int bourse;
00163 }inventaire_t;
00164
00166 typedef struct team_s{
00167     int nbPerso;
00168     perso_t *team[NB_TEAM_PERSOS];
00169 }team_t;
00170
00172 int creer_perso(perso_t *,char *, char *,SDL_Renderer*,float * statPerso);
00173
00174 void lvl_up(perso_t*,float * statPerso);
00175 void aff_perso(perso_t perso);
```

```
00176
00177 void equiper_stuff(equipement_t* equipement, perso_t* perso);
00178 int desequiper_stuff(int iStuff, perso_t* perso);
00179 void utiliser_obj(inventaire_t *,perso_t *,int);
00180
00182 int join_team(team_t *,perso_t *);
00183 void init_reinit__team(team_t *);
00184
00186 void creer_all_objet(objet_t *, SDL_Renderer* );
00187
00189 attaque_t creer_attaque(char nom[TAILLE_NOM],type_att_t type_att, char desc[100], effet_t effet, int
      qte_effet, int duree_effet, int nb_cible, int precision);
00190
00192 void creer_all_sprite_equipement(SDL_Texture* allSprite[ALL_EQUIPEMENTS], SDL_Renderer* ecran);
00193 void creer_all_equipement(equipement_t allStuff[ALL_EQUIPEMENTS][3],SDL_Texture*
      allSprite[ALL_EQUIPEMENTS]);
00194
00195
00197 void init_and_reinit_inv(inventaire_t * inv);
00198 int jeter_obj(inventaire_t *,int );
00199 int ajouter_obj(inventaire_t *,objet_t *,int );
00200 int ajouter_stuff(equipement_t allStuff[ALL_EQUIPEMENTS][3],inventaire_t *,int iStuff);
00201 int jeter_stuff(inventaire_t *Inv,int iStuff);
00202
00204 void afficher_arme(arme_t* arme);
00205 void creer_all_arme(arme_t armes[ALL_ARMES][3],SDL_Renderer* ecran);
00206 void creer_all_att_arme(arme_t armes[ALL_ARMES][3]);
00207 void equiper_arme(arme_t* arme, perso_t* perso);
00208 int desequiper_arme(arme_t * arme, perso_t* perso);
00209 int ajouter_arme(arme_t allArmes[ALL_ARMES][3],inventaire_t *inv,int indArm);
00210 int jeter_arme(inventaire_t *Inv,arme_t* arme);
00211
00213 mob_t* creer_mob(char * nom,SDL_Renderer* ecran,int lvl);
00214 #endif
```

4.18 lib/utils.h File Reference

programme gérant les fonctions pour les sauvegardes et l'entrée du pseudo du joueur.

```
#include <stdio.h>
#include <string.h>
#include <sys/stat.h>
#include ¨../lib/type.h¨
#include ¨../lib/texture.h¨
#include ¨../lib/fmap.h¨
```

Macros

- #define **WINDOW_WIDTH** 1360
- #define **WINDOW_HEIGHT** 780
- #define **LETTER_WIDTH** 20
- #define **LETTER_HEIGHT** 50

Functions

- void **entrerLettre** (SDL_Event event, char ∗name, int ∗indice, SDL_Color color, int ∗quit)

  Fonction de détection de chaque touche utile du clavier pour entrer un nom : condition qui permet de vérifier si la longueur est bien inférieure à la taille du nom maximale (TAILLE_NOM - 1) pour pouvoir insérer un caractère condition qui permet de vérifier si la longueur est bien supérieure à 0 pour pouvoir supprimer un caractère condition qui permet de vérifier si la longueur est bien supérieure à 1 pour pouvoir valider le nom si et seulement si l'indice pointe sur le début de la chaîne, la lettre sera en majuscule.
- char ∗ entrerNom (SDL_Renderer ∗renderer)

  Fonction d'entrée du nom.

- void sauvegarder (int nb_save, int idMap, int x, int y, team_t team, inventaire_t inv, int lorePos, objet_t allPot[ALL_OBJETS], equipement_t allStuff[ALL_EQUIPEMENTS][3], arme_t allArmes[ALL_ARMES][3], SDL_Renderer *renderer, SDL_Texture *Map, SDL_Rect dest)

  Fonction de sauvegarde de la progression.
- void chargerSave (int nb_save, int *idMap, int *x, int *y, team_t *team, inventaire_t *inv, int *lorePos, objet_t allPot[ALL_OBJETS], equipement_t allStuff[ALL_EQUIPEMENTS][3], perso_t allPerso[NB_TEAM_PERSOS], arme_t allArmes[ALL_ARMES][3], float stats[NB_TEAM_PERSOS][8], SDL_Renderer *renderer)

  Fonction de chargement de la sauvegarde.
- void nouvelleSave (int nb_save, char *nom)

  Fonction d'initialisation de la sauvegarde.

4.18.1 Detailed Description

programme gérant les fonctions pour les sauvegardes et l'entrée du pseudo du joueur.

Author

Warrick Bonga.

Version

1.9.8

Date

16 Avril 2024.

4.18.2 Function Documentation

4.18.2.1 chargerSave()

```
void chargerSave (
            int nb_save,
            int * idMap,
            int * x,
            int * y,
            team_t * team,
            inventaire_t * inv,
            int * lorePos,
            objet_t allPot[ALL_OBJETS],
            equipement_t allStuff[ALL_EQUIPEMENTS][3],
            perso_t allPerso[NB_TEAM_PERSOS],
            arme_t allArmes[ALL_ARMES][3],
            float stats[NB_TEAM_PERSOS][8],
            SDL_Renderer * renderer )
```

Fonction de chargement de la sauvegarde.

Ouverture du fichier contenant la sauvegarde

Déclaration des indices, de l'élément courant et du caractère courant lors de la lecture du fichier

Déclaration des variables provisoires permettant la lecture des statistiques

Chargement de la dernière carte du monde sauvegardée ainsi que la position du personnage dessus

Chargement des personnages et de leurs statistiques

Chargement des objets de l'inventaire (potions)

Chargement de l'équipement de l'inventaire (armure) et de la bourse (montant de la monnaie du jeu)

Chargement des armes de l'inventaire

Chargement de l'avancement dans l'histoire du jeu

Fermeture du fichier contenant la sauvegarde

4.18.2.2 entrerNom()

```
char * entrerNom (
            SDL_Renderer * renderer )
```

Fonction d'entrée du nom.

Déclaration des variables d'event et d'arrêt de la boucle

Déclaration des chaînes de caractères de la fenêtre (en-tête et règles d'entrée du nom)

Déclaration de la couleur blanche avec le type personalisé de SDL

Création des textures des textes déclarés plus haut

Création et allocation des zones des textures + copie dans renderer

Déclaration du nom et de l'indice d'écriture dans le nom

Premier bourrage de la chaîne du nom pour éviter un affichage de caractères aléatoires

Boucle d'entrée du nom

L'entrée de lettres s'activera au moment de l'appui sur une des touches disponibles (cf. fonction entrerLettre)

Pour éviter une erreur lorsque la largeur de la texture vaut 0, le premier caractère du nom vaut '_' lorsque celui-ci est vide

Écriture du nom sur l'écran

Rafraîchissement de l'écran

Libération en mémoire des textures

Appuyer sur échap retourne le nom ¨Golem¨ par défaut

Bourrage de la place allouée non utilisée

Renvoi de l'adresse du nom

4.18.2.3 nouvelleSave()

```
void nouvelleSave (
            int nb_save,
            char * nom )
```

Fonction d'initialisation de la sauvegarde.

Création du dossier contenant les sauvegardes

Ouverture du fichier d'initialisation de la sauvegarde

Initialisation de la première carte du monde ainsi que la position du personnage au départ

Initialisation du personnage de départ et de ses statistiques

Intialisation de l'emplacement des objets de l'inventaire (potions)

Intialisation de l'emplacement de l'équipement de l'inventaire (armure) et de la bourse (montant de la monnaie du jeu)

Initialisation de l'emplacement des armes de l'inventaire

Initialisation de la valeur représentant l'avancement dans l'histoire du jeu

Fermeture du fichier d'initialisation de la sauvegarde

4.18.2.4 sauvegarder()

```
void sauvegarder (
            int nb_save,
            int idMap,
            int x,
            int y,
            team_t team,
            inventaire_t inv,
            int lorePos,
            objet_t allPot[ALL_OBJETS],
            equipement_t allStuff[ALL_EQUIPEMENTS][3],
            arme_t allArmes[ALL_ARMES][3],
            SDL_Renderer * renderer,
            SDL_Texture * Map,
            SDL_Rect dest )
```

Fonction de sauvegarde de la progression.

Déclaration de l'image pour la fenêtre de la sauvegarde

Déclaration des images textuelles pendant et après la sauvegarde

Ouverture du fichier contenant la sauvegarde

Déclaration des indices

Affichage du texte pendant la sauvegarde

Sauvegarde de la dernière carte du monde chargée ainsi que la position du personnage dessus

Sauvegarde des personnages et de leurs statistiques

Sauvegarde des objets de l'inventaire (potions)

Sauvegarde de l'équipement de l'inventaire (armure) et de la bourse (montant de la monnaie du jeu)

Sauvegarde des armes de l'inventaire

Sauvegarde de l'avancement dans l'histoire du jeu

Affichage du texte après la sauvegarde

Affichage du texte jusqu'à qu'un clic de souris soit détecté

Destruction des textures

Fermeture du fichier contenant la sauvegarde

4.19 utils.h

Go to the documentation of this file.

```
00001
00009 #include <stdio.h>
00010 #include <string.h>
00011 #include <sys/stat.h>
00012 #include "../lib/type.h"
00013 #include "../lib/texture.h"
00014 #include "../lib/fmap.h"
00015
00016 #define WINDOW_WIDTH 1360
00017 #define WINDOW_HEIGHT 780
00018
00019 #define LETTER_WIDTH 20
00020 #define LETTER_HEIGHT 50
00021
00022 void entrerLettre(SDL_Event event, char * name, int * indice,  SDL_Color color, int * quit);
00023 char * entrerNom(SDL_Renderer * renderer);
00024 void sauvegarder(int nb_save, int idMap, int x, int y, team_t team, inventaire_t inv, int lorePos,
      objet_t allPot[ALL_OBJETS], equipement_t allStuff[ALL_EQUIPEMENTS][3], arme_t allArmes[ALL_ARMES][3],
      SDL_Renderer * renderer,SDL_Texture * Map, SDL_Rect dest);
00025 void chargerSave(int nb_save, int *idMap, int *x, int *y, team_t * team, inventaire_t * inv,int
      *lorePos, objet_t allPot[ALL_OBJETS], equipement_t allStuff[ALL_EQUIPEMENTS][3], perso_t
      allPerso[NB_TEAM_PERSOS], arme_t allArmes[ALL_ARMES][3], float stats[NB_TEAM_PERSOS][8], SDL_Renderer
      * renderer);
00026 void nouvelleSave(int nb_save, char * nom);
```

4.20 src/combat.c File Reference

programme gérant les combats

```
#include ¨../lib/EventOp.h¨
#include ¨../lib/combat.h¨
#include <string.h>
#include <time.h>
#include <stdlib.h>
```

Functions

- int degats_mob (mob_t ∗mob, attaque_t att, perso_t ∗perso)

  fonctions de calcul des degats
- int **degats_perso** (perso_t ∗perso, attaque_t att, mob_t ∗mob)

  dégâts infligés par les mobs
- void **fin_combat** (int vict, mob_t ∗mob, inventaire_t ∗inv, team_t ∗team, float statPerso[3][8], scene_t ∗scene, int ∗click, int ∗over)
- void DetectEventsFight (int ∗menu_combat, int ∗MouseClick, int ∗MouseOver, int ∗quit, int ∗perso, int ∗obj, scene_t ∗scene, int w, int h, int ∗W, int ∗H, int ∗atta)

  fonction DetectEventsFight qui s'occupera de gérer la détection des événements quand nous sommes en combat
- void ReactEventsFight (int ∗p, int perso, int obj, int MouseClick, int MouseOver, int ∗menu_combat, SDL_↩ Renderer ∗pRenderer, SDL_Texture ∗fond_combat, int w, int h, inventaire_t ∗inv, team_t ∗team, int imob, int H, int W, int atta, float statPerso[3][8], scene_t ∗scene, int ∗pvMob)

  fonction ReactEventsInvSac qui s'occupe de réagir en fonction des événements quand nous sommes sur le menu d'inventaire sur le sac

4.20.1 Detailed Description

programme gérant les combats

Author

Lenny Borry.

Version

1.9.5

Date

16 Avril 2024.

4.20.2 Function Documentation

4.20.2.1 degats_mob()

```
int degats_mob (
            mob_t * mob,
            attaque_t att,
            perso_t * perso )
```

fonctions de calcul des degats

dégâts infligés par les mobs

4.20.2.2 DetectEventsFight()

```
void DetectEventsFight (
             int * menu_combat,
             int * MouseClick,
             int * MouseOver,
             int * quit,
             int * perso,
             int * obj,
             scene_t * scene,
             int w,
             int h,
             int * W,
             int * H,
             int * atta )
```

fonction DetectEventsFight qui s'occupera de gérer la détection des événements quand nous sommes en combat

création de la variable event qui récupère les événements

coordonnés du rectangle d'options en combat

largeur du grand rectangle

largeur des 4 carrés qui coupent le grand rectangle

longueur du rectangle indiquant le nom des persos

largeur du rectangle indiquant le nom des persos

longueur des rectangles ¨attaque¨ et ¨item¨

largeur des rectangles ¨attaque¨ et ¨item¨

longueur des rectangles de chaque attaque

largeur des rectangles de chaque attaque

longueur des rectangles de chaque objet

largeur des rectangles de chaque objet

longueur du rectangle du nom de l'attaque

largeur du rectangle du nom de l'attaque

longueur des rectangles ¨puissance¨ et ¨précision¨

largeur des rectangles ¨puissance¨ et ¨précision¨

taille du carré dans lequel il y aura le logo représentant le type de l'attaque

longueur du rectangle de description de l'attaque

largeur du rectangle de description de l'attaque

longueur du rectangle du nom de l'objet

largeur du rectangle du nom de l'objet

taille du carré du pixel art de l'objet

longueur du rectangle de description de l'objet

largeur du rectangle de description de l'objet

longueur du rectangle du bouton ¨utiliser¨

largeur du rectangle du bouton ¨utiliser¨

écart vertical entre les boutons indiquant le nom des persos

écart horizontal entre les boutons indiquant le nom des persos

écart vertical entre les boutons ¨attaque¨ et ¨item¨

écart horizontal entre les boutons ¨attaque¨ et ¨item¨

écart vertical entre les rectangles de chaque attaque

écart horizontal entre les rectangles de chaque attaque

écart vertical entre les rectangles de chaque objet

écart horizontal entre les rectangles de chaque objet

écart vertical entre les rectangles du nom de l'attaque et de description de l'attaque

écart horizontal entre les rectangles du nom de l'attaque, de la puissance et du logo

écart vertical entre les rectangles ¨puissance¨ et ¨précision¨

écart vertical entre les rectangles du logo et de description

écart vertical entre les rectangles du nom, du pixel art, de la description et du bouton ¨utiliser¨ de l'objet

écart horizontal de chaque côté du rectangle nom de l'objet

écart horizontal de chaque côté du pixel art de l'objet

écart horizontal de chaque côté du rectangle description de l'objet

écart horizontal de chaque côté du bouton ¨utiliser¨

taille du carré du bouton retour

écart entre le bouton retour et le bord

boucle d'attente d'événements

regarde le bouton pour quitter la fenêtre si appuyer met quit à 1 ce qui va arrêter la boucle de jeu

récupération des coordonnées de la souris

regarde si la souris est dans la case des persos

rectangle perso 1

rectangle perso 2

rectangle perso 3
regarde si la souris est dans la case de attaque ou item
rectangle attaque
rectangle item
test si la souris est sur le bouton retour
regarde si la souris est dans la case des attaques
rectangle attaque 1
rectangle attaque 2
rectangle attaque 3
rectangle attaque 4
test si la souris est sur le bouton retour
regarde si la souris est dans la case de description d'attaque
rectangle nom de l'attaque
rectangle description
rectangle puissance
rectangle précision
rectangle description
rectangle du logo
rectangle description
test si la souris est sur le bouton retour
regarde si la souris est dans la case des objets
rectangle objet 1
rectangle objet 2
rectangle objet 3
rectangle objet 4
rectangle objet 5
rectangle objet 6
rectangle objet 7
rectangle objet 8
rectangle objet 9
rectangle objet 10
test si la souris est sur le bouton retour
regarde si la souris est dans la case de description d'objet
rectangle nom de l'objet
rectangle pixel art de l'objet
rectangle description
rectangle utiliser
test si la souris est sur le bouton retour
lancer l'attaque
utiliser l'item
récupération si le bouton de la souris est lâché
regarde si le bouton est relâché sur un des boutons

4.20.2.3 ReactEventsFight()

```
void ReactEventsFight (
            int * p,
            int perso,
            int obj,
            int MouseClick,
            int MouseOver,
            int * menu_combat,
            SDL_Renderer * pRenderer,
            SDL_Texture * fond_combat,
            int w,
            int h,
            inventaire_t * inv,
            team_t * team,
            int imob,
            int H,
            int W,
            int atta,
            float statPerso[3][8],
            scene_t * scene,
            int * pvMob )
```

fonction ReactEventsInvSac qui s'occupe de réagir en fonction des événements quand nous sommes sur le menu d'inventaire sur le sac

déclaration de i servant aux boucles for et textwidth, textheight pour les dimensions des textes

longueur du rectangle indiquant le nom des persos

largeur du rectangle indiquant le nom des persos

longueur des rectangles ¨attaque¨ et ¨item¨

largeur des rectangles ¨attaque¨ et ¨item¨

longueur des rectangles de chaque attaque

largeur des rectangles de chaque attaque

longueur des rectangles de chaque objet

largeur des rectangles de chaque objet

longueur du rectangle du nom de l'attaque

largeur du rectangle du nom de l'attaque

longueur des rectangles ¨puissance¨ et ¨précision¨

largeur des rectangles ¨puissance¨ et ¨précision¨

taille du carré dans lequel il y aura le logo représentant le type de l'attaque

longueur du rectangle de description de l'attaque

largeur du rectangle de description de l'attaque

longueur du rectangle du nom de l'objet

largeur du rectangle du nom de l'objet

taille du carré du pixel art de l'objet

longueur du rectangle de description de l'objet

largeur du rectangle de description de l'objet

longueur du rectangle du bouton ¨utiliser¨

largeur du rectangle du bouton ¨utiliser¨

écart vertical entre les boutons indiquant le nom des persos

écart horizontal entre les boutons indiquant le nom des persos

écart vertical entre les boutons ¨attaque¨ et ¨item¨

écart horizontal entre les boutons ¨attaque¨ et ¨item¨

écart vertical entre les rectangles de chaque attaque

écart horizontal entre les rectangles de chaque attaque

écart vertical entre les rectangles de chaque objet

écart horizontal entre les rectangles de chaque objet

écart vertical entre les rectangles du nom de l'attaque et de description de l'attaque

écart horizontal entre les rectangles du nom de l'attaque, de la puissance et du logo

écart vertical entre les rectangles du logo et de description

écart vertical entre les rectangles ¨puissance¨ et ¨précision¨

écart vertical entre les rectangles du nom, du pixel art, de la description et du bouton ¨utiliser¨ de l'objet

écart horizontal de chaque côté du rectangle nom de l'objet

écart horizontal de chaque côté du pixel art de l'objet

écart horizontal de chaque côté du rectangle description de l'objet

écart horizontal de chaque côté du bouton ¨utiliser¨

taille du carré du bouton retour

écart entre le bouton retour et le bord

chaîne de caractères utilisée pour l'affichage des persos avec leur nb de pv et pm

chaîne de caractères pour la conversion d'entiers en caractères

déclaration et calcul des variables des coordonnées des carrés des persos et des mobs

taille du carré du premier perso de la team

taille du carré du 2e perso de la team

taille du carré du 3e perso de la team

écart vertical entre les persos de la team

écart horizontal à gauche du premier perso et à droite du troisieme perso

écart horizontal à droite du premier perso

écart horizontal à gauche et à droite du deuxieme perso

écart horizontal à gauche du troisieme perso

taille du carré du mob

écart vertical au dessus et au dessus du mob

écart horizonral à gauche et à droite du mob

longueur du rectangle du nom des pv

largeur du rectangle du nom des pv

écart horizontal à gauche du 3e perso

écart horizontal à gauche du 2e perso

écart horizontal à gauche du 1er perso

écart vertical au dessus des pv du 3e perso

écart vertical au dessus des pv du 2e perso

écart vertical au dessus des pv du 1er perso

écart vertical entre les pv et le mob

mise en place des coordonnées dans les tableaux correspondants

chargement des textures des boutons de fond et leur version hover (lorsque la souris est dessus)

chargement des textures du bouton retour et de sa version hover (lorsque la souris est dessus)

déclaration des rect correspondant aux coordonnées et dimensions d'affichage de toute l'interface de combat

grand rectangle qui va contenir tous les rectangles ci-dessous

rectangles des persos

rectangles attaque et item

rectangles de chaque objet

rectangles de chaque attaque

rectangles du détail de chaque attaque

rectangles du détail de chaque objet

rectangles de retour

déclaration de la couleur du texte

affichage des fonds à l'écran

déclaration et initialisation des rectangles et des textures de chaque perso et de chaque mob

chaque 'case' représente le n-ième perso. S'il y a 3 persos, je crée le rectangle des 3 persos, s'il y en a 2 j'en crée que 2 etc

de même pour les textures, je charge leur IdleAnim (lorsque le perso bouge sur lui-même)

on se décale d'une image dans le sprite de l'Idle Animation de tous les persos et mobs

déclaration et initialisation des textures des textes de chaque bouton

nom des persos

texture erreur

textures attaque et item

textures de chaque attaque

matrices contenant les 4 attaques de chacun des persos de la team

déclaration et initialisation des textures des objets

lance l'attaque

calcul précision de l'attaque

utilise l'objet

affichage de tous les rectangles selon menu_combat

destruction des textures

délai d'affichage

4.21 src/EventOp.c File Reference

programme gérant les events dans l'open-world

```
#include ¨../lib/EventOp.h¨
#include <time.h>
```

Functions

- void transitionFondu (SDL_Renderer *renderer, int largeur_ecran, int hauteur_ecran, int duree)

  Fonction pour réaliser une transition en fondu.
- void **chargement** (scene_t *scene, SDL_Renderer *pRenderer, scene_t futurScene)

  fonction faisant l'écran de chargement entre certaines scènes

- void cinematic_start (SDL_Renderer *pRenderer, int *next, SDL_Rect *P1, SDL_Rect *P2, SDL_Rect *P3, SDL_Rect *P4, scene_t *scene, SDL_Texture **storyP1, SDL_Texture **storyP2, SDL_Texture **storyP3, SDL_Texture **storyP4, int *skipOn, SDL_Texture **Map1, SDL_Texture **Map2, SDL_Texture **Map3, SDL_Texture **Map4, SDL_Texture **Map5, SDL_Texture **Map6, int colli[125][250])

  fonction gérant l'affichage et les événements de la cinématique de début de jeu
- void DetectEventsOp (int *MouseOver, int *KeyIsPressed, int *Action, scene_t *scene, int w, int h, int *choiceShop, int *vendeur, int *numperso)

  fonction DetectEventsOp qui s'occupera de gérer la détection des événements quand nous sommes sur l'openWorld
- void chargement_barre_pv (int pourcent, SDL_Texture **pvJauge, SDL_Texture **pvActu, SDL_Renderer *pRenderer)

  fonction chargement_barre_pv qui s'occupe de charger la bonne texture pour la barre de PV selon le % de PV actuel
- void ReactEventsOp (int *avancement_quete, int MouseOver, int KeyIsPressed, int *Action, SDL_Renderer *pRenderer, SDL_Texture *Perso, SDL_Rect marcheHori[3][8], SDL_Rect *dest, int *i, int w, int h, team_t team, int biome, int choiceShop, int *vendeur, int colli[125][250], SDL_Texture *vendeurDialoTexture[5], int *map, SDL_Rect *savedest, int numperso)

  fonction ReactEventsOp qui s'occupe de réagir en fonction des événements quand nous sommes sur l'openWorld
- void DetectEventsInvSac (int *useOk, int *yonUse, int *yonThrow, int *MouseClick, int *MouseOver, scene_t *scene, int w, int h, SDL_Rect destThrowButton)

  fonction DetectEventsInvSac qui s'occupera de gérer la détection des événements quand nous sommes sur le menu d'inventaire sur le sac
- void ReactEventsInvSac (int *useOk, int *yonUse, int *yonThrow, int MouseClick, int KeyIsPressed, int MouseOver, SDL_Renderer *pRenderer, int w, int h, inventaire_t *inv, SDL_Rect *destThrowButton, perso_t **team)

  fonction ReactEventsInvSac qui s'occupe de réagir en fonction des événements quand nous sommes sur le menu d'inventaire sur le sac
- void DetectEventsStats (int *numPage, int *MouseClick, int *MouseOver, scene_t *scene, int w, int h, SDL↩_Rect *destStatAction, team_t *team, float Stats[3][8])

  fonction DetectEventsStats qui s'occupera de gérer la détection des événements quand nous sommes sur le menu des stats des personnages
- void ReactEventsStats (int numPage, int MouseClick, int KeyIsPressed, int MouseOver, SDL_Renderer *p↩Renderer, int w, int h, team_t *team, float Stats[3][8], SDL_Rect *destStatAction)

  fonction ReactEventsInvSac qui s'occupe de réagir en fonction des événements quand nous sommes sur le menu d'inventaire sur le sac
- void DetectEventsInvStuff (int *yonThrow, int *MouseClick, int *MouseOver, scene_t *scene, int w, int h, SDL_Rect destThrowButton, int *numPage, int *numPagePerso, int *slot, int *eOuStuff)

  fonction DetectEventsInvStuff qui s'occupera de gérer la détection des événements quand nous sommes sur le menu d'inventaire pour les équipements/armes
- void ReactEventsInvStuff (int *yonThrow, int MouseClick, int MouseOver, SDL_Renderer *pRenderer, int w, int h, inventaire_t *inv, SDL_Rect *destThrowButton, team_t *team, int numPage, int numPagePerso, int slot, int *eOuStuff)

  fonction ReactEventsInvStuff qui s'occupe de réagir en fonction des événements quand nous sommes sur le menu d'inventaire pour les équipements/armes
- int achat_possible (int choiceBiy, int biome, inventaire_t *inv, objet_t allPot[ALL_OBJETS], equipement_t allStuff[ALL_EQUIPEMENTS][3], arme_t armes[ALL_ARMES][3])

  vérifie si on a assez d'argent pour acheter l'item sélectionné, si on a la place pour le stocker ou si on n'a pas 3 fois le même équipement si les conditions sont réunies, on ne touche pas à achatPossible, sinon si l'une des vérifications ne vaut pas vrai on met achatPossible à 0

- void EventShop (int ∗MouseOver, int ∗Action, SDL_Renderer ∗pRenderer, int biome, equipement_t all↩ Stuff[ALL_EQUIPEMENTS][3], inventaire_t ∗inv, int w, int h, scene_t ∗scene, objet_t allPot[ALL_OBJETS], int ∗choiceBiy, arme_t armes[ALL_ARMES][3])

  fonction EventsInvShop qui gère le menu d'achat dans le magasin pour acheter équipement, armes et objets
- void EventSell (int ∗MouseOver, int ∗MouseClick, int ∗Action, SDL_Renderer ∗pRenderer, inventaire_t ∗inv, int w, int h, scene_t ∗scene, int ∗numPage)

4.21.1 Detailed Description

programme gérant les events dans l'open-world

Author

Cognard Luka,François Lépine.

Version

1.9.8

Date

16 Avril 2024.

4.21.2 Function Documentation

4.21.2.1 achat_possible()

```
int achat_possible (
            int choiceBiy,
            int biome,
            inventaire_t * inv,
            objet_t allPot[ALL_OBJETS],
            equipement_t allStuff[ALL_EQUIPEMENTS][3],
            arme_t armes[ALL_ARMES][3] )
```

vérifie si on a assez d'argent pour acheter l'item sélectionné, si on a la place pour le stocker ou si on n'a pas 3 fois le même équipement si les conditions sont réunies, on ne touche pas à achatPossible, sinon si l'une des vérifications ne vaut pas vrai on met achatPossible à 0

cas où magasin du biome 1 à 3

cas où magasin du biome 5

cas où magasin du biome 4

cas où magasin du biome 1 à 3

cas où magasin du biome 5

cas où magasin du biome 4

objets

cas différents selon chaque biome

biome 1

biome 2

biome 3

biome 4

biome 5

4.21.2.2 chargement_barre_pv()

```
void chargement_barre_pv (
            int pourcent,
            SDL_Texture ** pvJauge,
            SDL_Texture ** pvActu,
            SDL_Renderer * pRenderer )
```

fonction chargement_barre_pv qui s'occupe de charger la bonne texture pour la barre de PV selon le % de PV actuel

barre verte si le nombre de PV est supérieur ou égal à 50%

barre jaune si le nombre de PV est supérieur ou égal à 20% mais est inférieur à 50%

barre rouge si le nombre de PV est inférieur à 20%

4.21.2.3 cinematic_start()

```
void cinematic_start (
            SDL_Renderer * pRenderer,
            int * next,
            SDL_Rect * P1,
            SDL_Rect * P2,
            SDL_Rect * P3,
            SDL_Rect * P4,
            scene_t * scene,
            SDL_Texture ** storyP1,
            SDL_Texture ** storyP2,
            SDL_Texture ** storyP3,
            SDL_Texture ** storyP4,
            int * skipOn,
            SDL_Texture ** Map1,
            SDL_Texture ** Map2,
            SDL_Texture ** Map3,
            SDL_Texture ** Map4,
            SDL_Texture ** Map5,
            SDL_Texture ** Map6,
            int colli[125][250] )
```

fonction gérant l'affichage et les événements de la cinématique de début de jeu

création de la variable event qui récupère les événements

boucle d'attente d'événement

récupération des touchess relâchées

regarde si la touche entrée a été relâchée

récupération de la position de la souris

regarde si la souris est sur le bouton skip

récupération d'un bouton relâché venant de la souris

vérifie que la souris est sur le bouton skip pour réagir au clic

passe l'histoire, réinitialisation de toutes les variables utilisées ici puis passage sur l'écran de chargement vers l'OP

chargement des textures des map

texte de l'histoire séparée en 2 parties (ici 1ère partie)

texte de l'histoire séparée en 2 parties ici (2ème partie)

chargement des textures des textes

chargement de la texture, déclaration des dimensions et des coordonnées du bouton skip

déclaration des dimensions et des coordonnées des textes

affiche la première partie de l'histoire (texte numéro 1, puis 2 après avoir appuyé sur Entrée)

affiche la deuxième partie de l'histoire (même processus)

affiche la troisième partie de l'histoire (même processus)

affiche la quatrième partie de l'histoire (même processus)

fin de l'histoire, réinitialisation de toutes les variables utilisées ici puis passage sur l'écran de chargement vers l'OP

destruction des images illustrant l'histoire

chargement des textures des map

destruction de toutes les textures créées

délai d'affichage

4.21.2.4 DetectEventsInvSac()

```
void DetectEventsInvSac (
             int * useOk,
             int * yonUse,
             int * yonThrow,
             int * MouseClick,
             int * MouseOver,
             scene_t * scene,
             int w,
             int h,
             SDL_Rect destThrowButton )
```

fonction DetectEventsInvSac qui s'occupera de gérer la détection des événements quand nous sommes sur le menu d'inventaire sur le sac

création de la variable event qui récupère les événements

coordonnés des zones ou se situent les zones d'objet dans le menu

boucle d'attente d'événement

récupération des touches enfoncées

regarde quelle touche est enfoncée

récupération des coordonnées de la souris

regarde si la souris est dans les cases d'objet

regarde si la souris est sur le bouton jeter

regarde si la souris est sur le bouton utiliser

regarde si la souris est sur le bouton oui, non ou sur aucun des deux de la boîte de texte pour jeter l'objet

regarde si la souris est sur un personnage pouvant utiliser un objet

récupération si le bouton de la souris est lâché

regarde si le bouton est relâché sur une zone d'objet de l'inventaire

regarde si le bouton est relâché sur le bouton jeter

regarde si le bouton est relâché sur le bouton oui ou non de la boîte de texte pour jeter l'objet

regarde si le bouton est relâché sur le bouton utiliser

4.21.2.5 DetectEventsInvStuff()

```
void DetectEventsInvStuff (
             int * yonThrow,
             int * MouseClick,
             int * MouseOver,
             scene_t * scene,
             int w,
             int h,
             SDL_Rect destThrowButton,
             int * numPage,
             int * numPagePerso,
             int * slot,
             int * eOuStuff )
```

fonction DetectEventsInvStuff qui s'occupera de gérer la détection des événements quand nous sommes sur le menu d'inventaire pour les équipements/armes

création de la variable event qui récupère les événements

coordonnés des zones ou se situent les zones d'objet dans le menu

coordonnés du personnage affiché

boucle d'attente d'événement

récupération des touches relâchées

regarde quelle touche est relâchée

réinitialise toutes les variables utiliser en même temps de changer de scene

récupération des coordonnées de la souris

regarde si la souris est dans les cases des équipements/armes

regarde si la souris est dans un slot d'équipement

slot 1 et 2

slot 3 et 4

slot 5 et 6

regarde si la souris est sur le bouton équiper ou déséquiper

bouton équiper

bouton déséquiper

si la souris n'est sur aucun bouton

regarde si la souris est sur le bouton jeter

regarde si la souris est sur le bouton flèche droite ou gauche dans l'inventaire

flèche gauche

flèche droite

regarde si la souris est sur le bouton flèche droite ou gauche sur les personnages

flèche gauche

flèche droite

regarde si la souris est sur le bouton oui, non ou sur aucun des deux de la boîte de texte pour jeter l'objet

récupération si le bouton de la souris est lâché

regarde si le bouton est relâché sur une zone d'objet de l'inventaire

regarde si le bouton est relâché sur le bouton équiper ou déséquiper

regarde si le bouton est relâché sur un slot d'équipements/armes

regarde si le bouton est relâché sur le bouton jeter

regarde si le bouton est relâché sur le bouton oui ou non de la boîte de texte pour jeter l'objet

4.21.2.6 DetectEventsOp()

```
void DetectEventsOp (
             int * MouseOver,
             int * KeyIsPressed,
             int * Action,
             scene_t * scene,
             int w,
             int h,
             int * choiceShop,
             int * vendeur,
             int * numperso )
```

fonction DetectEventsOp qui s'occupera de gérer la détection des événements quand nous sommes sur l'openWorld

création de la variable event qui récupère les événements

boucle d'attente d'événement

récupération des touchess enfoncées

regarde quelle touche est enfoncée

change la variable KeyIsPressed si la touche est maintenue

récupération des coordonnées de la souris

regarde si la souris est dans les cases d'objet

récupération des touchess relâchées et changement de KeyIsPressed

regarde quelle touche a été relâchée et change la variable KeyIsPressed en fonction de la touche

touche de déplacement z, q, s, d

touche pour changer de personnage

touche d'interaction

touche pour faire pause

touche d'ouverture d'inventaire (sac)

touche d'ouverture d'inventaire (équipement et armes)

touche d'ouverture des stats

touche flèche du haut pour les choix du vendeur

touche flèche du bas pour les choix du vendeur

touche entrée pour valider les choix du vendeur

détection clic souris relaché

icône d'inventaire (sac d'objet)

icône des stats

icône d'inventaire (équipement et armes)

4.21.2.7 DetectEventsStats()

```
void DetectEventsStats (
            int * numPage,
            int * MouseClick,
            int * MouseOver,
            scene_t * scene,
            int w,
            int h,
            SDL_Rect * destStatAction,
            team_t * team,
            float Stats[3][8] )
```

fonction DetectEventsStats qui s'occupera de gérer la détection des événements quand nous sommes sur le menu des stats des personnages

création de la variable event qui récupère les événements

boucle d'attente d'événement

récupération des touches enfoncées

regarde quelle touche est enfoncée

récupération des coordonnées de la souris

regarde si la souris est sur les flèches pour changer de perso sur le bouton valider

regarde si la souris est sur un bouton +

regarde si la souris est sur un bouton -

regarde si la souris est dans la case + ou - d'une stat

récupération si le bouton de la souris est lâché

regarde si le bouton est relâché sur une zone d'objet de l'inventaire

4.21.2.8 EventSell()

```
void EventSell (
            int * MouseOver,
            int * MouseClick,
            int * Action,
            SDL_Renderer * pRenderer,
            inventaire_t * inv,
            int w,
            int h,
            scene_t * scene,
            int * numPage )
```

déclaration du nombre de pages et des variables pour la taille du texte en largeur et hauteur

création de la variable event qui récupère les événements

boucle d'attente d'événement

regarde la position de la souris

bouton retour (55) et vendre (54)

regarde si la souris est dans les cases d'équipements/armes

regarde si la souris est sur la flèche gauche ou droite

retour aux choix du vendeur si le bouton retour est actionné

bouton vendre actionné seulement si un équipement ou une arme est sélectionné

MouseClick prend la valeur de l'équipement ou de l'arme sélectionné dans l'inventaire

change de page selon la flèche sur laquelle on est

vérification du numéro de page pour éviter de dépasser le nombre de pages ou d'être en-dessous de 0

déclaration des textures

déclaration des couleurs

déclaration rect

déclaration char pour les prix ainsi que la bourse actuelle

chargement de la texture du texte bourse avec l'argent possédé

change l'affichage du bouton retour selon si on est dessus ou non

change l'affichage du bouton vendre selon si on est dessus ou non, si un équipement ou une arme est sélectionné ou non

affichage des bouton acheter et retour, de la zone d'item et de la bourse actuelle en haut à gauche

chargement des textures des noms, descriptions et des prix des équipements/armes dans l'inventaire

regarde si on est sur des équipements

regarde si on est sur des armes

sinon si l'inventaire a atteint sa limite mettre à NULL

affichage des équipements/armes et de la zone sur laquelle la souris a cliqué ou est positionnée

affiche seulement s'il y a un équipements/armes à cet emplacement

affichage de la zone autour des noms selon s'il est sélectionné ou non

mise à jour des coordonnées et dimensions des noms des équipements/armes

affichage des noms des équipements/armes

affichage de l'item sélectionné

affichage de l'image des équipements/armes ainsi que sa description

affichage flèche de gauche dans l'inventaire puis libère la mémoire

calcul des coordonnées/dimensions et affichage de la flèche droite dans l'inventaire puis libère la mémoire

libération de toutes les textures

4.21.2.9 EventShop()

```
void EventShop (
            int * MouseOver,
            int * Action,
            SDL_Renderer * pRenderer,
            int biome,
            equipement_t allStuff[ALL_EQUIPEMENTS][3],
            inventaire_t * inv,
            int w,
            int h,
            scene_t * scene,
            objet_t allPot[ALL_OBJETS],
            int * choiceBiy,
            arme_t armes[ALL_ARMES][3] )
```

fonction EventsInvShop qui gère le menu d'achat dans le magasin pour acheter équipement, armes et objets

création de la variable event qui récupère les événements

boucle d'attente d'événement

regarde la position de la souris

bouton retour (30) et acheter (29)

sur les items de la première étagère

sur les items de la deuxième étagère

sinon sur aucun bouton ou interaction

récupération d'un bouton relâché venant de la souris

bouton retour actionné on revient au choix du vendeur

toutes les réactions si le bouton acheter est actionné ou si un item est sélectionné

biome 1

item sélectionné

si le bouton acheter est cliqué et un objet est sélectionné

on vérifie que l'achat soit possible à chaque fois

cas d'un équipement

cas d'un objet

biome 4

item sélectionné

si le bouton acheter est cliqué et un objet est sélectionné

on vérifie que l'achat soit possible à chaque fois

cas d'un équipement

cas objet

biome 2,3 et 5

item sélectionné

si le bouton acheter est cliqué et un objet est sélectionné

on vérifie que l'achat soit possible à chaque fois

cas d'un équipement

biome 5

autre biome

cas objet

objets différents dans chaque magasin selon le biome

biome 2

biome 3

biome 5

déclaration texture

déclaration rect

déclaration couleur

déclaration char pour les prix ainsi que la bourse actuelle

change l'affichage du bouton retour selon si on est dessus ou non

change l'affichage du bouton acheter selon si on est dessus ou non, si on a l'argent nécessaire pour l'acheter et si on peut le stocker

affichage des boutons acheter et retour, de la zone d'item et de la bourse actuelle en haut à gauche

calcul des coordonnées et affichage des items ainsi que leur prix juste en-dessous

affiche le prix en blanc par défaut, et l'affiche en jaune si l'item est sélectionné

affiche le prix en blanc par défaut, et l'affiche en jaune si l'item est sélectionné

affiche le prix en blanc par défaut, et l'affiche en jaune si l'item est sélectionné

affiche le prix en blanc par défaut, et l'affiche en jaune si l'item est sélectionné

affiche le prix en blanc par défaut, et l'affiche en jaune si l'item est sélectionné

affiche le prix en blanc par défaut, et l'affiche en jaune si l'item est sélectionné

affiche le prix en blanc par défaut, et l'affiche en jaune si l'item est sélectionné

affiche le prix en blanc par défaut, et l'affiche en jaune si l'item est sélectionné

affiche le prix en blanc par défaut, et l'affiche en jaune si l'item est sélectionné

affiche le prix en blanc par défaut, et l'affiche en jaune si l'item est sélectionné

affiche le prix en blanc par défaut, et l'affiche en jaune si l'item est sélectionné

affiche le prix en blanc par défaut, et l'affiche en jaune si l'item est sélectionné

affiche le prix en blanc par défaut, et l'affiche en jaune si l'item est sélectionné

affiche le prix en blanc par défaut, et l'affiche en jaune si l'item est sélectionné

affiche le prix en blanc par défaut, et l'affiche en jaune si l'item est sélectionné

affiche le prix en blanc par défaut, et l'affiche en jaune si l'item est sélectionné

affiche le prix en blanc par défaut, et l'affiche en jaune si l'item est sélectionné

affiche le prix en blanc par défaut, et l'affiche en jaune si l'item est sélectionné

affiche le prix en blanc par défaut, et l'affiche en jaune si l'item est sélectionné

affiche le prix en blanc par défaut, et l'affiche en jaune si l'item est sélectionné

destruction des textures

délai d'affichage

4.21.2.10 ReactEventsInvSac()

```
void ReactEventsInvSac (
            int * useOk,
            int * yonUse,
            int * yonThrow,
            int MouseClick,
            int KeyIsPressed,
            int MouseOver,
            SDL_Renderer * pRenderer,
            int w,
            int h,
            inventaire_t * inv,
            SDL_Rect * destThrowButton,
            perso_t ** team )
```

fonction ReactEventsInvSac qui s'occupe de réagir en fonction des événements quand nous sommes sur le menu d'inventaire sur le sac

déclaration de i servant aux boucles for et textwidth, textheight pour les dimensions des textes

chaîne de caractère utilisée pour l'affichage des persos avec leur nb de pv et pm

chargement des textures nécessaires au menu

chargement des textures nécessaires au bouton jeter et sa boîte

chargement des textures nécessaires au bouton utiliser et sa boîte

déclaration des rect correspondant aux coordonnées et dimensions d'affichage du menu

déclaration des rect correspondant aux coordonnées et dimensions d'affichage des boîtes utiliser et jeter ainsi que des personnages pour la boîte d'utilisation

déclaration et initialisation des textures, des noms et descriptions des objets ainsi que la couleur des textes

déclaration et initialisation des textures des textes des personnages dans la boîte d'utilisation d'objet

déclaration de la variable pour les coordonnées et dimensions des noms d'objet

coordonnées et dimensions de la description des objets

coordonnées et destination du bouton jeter

coordonnées et destination du bouton utiliser

affichage du menu

affichage des objets présent ou non ainsi que la zone sur laquelle la souris a cliqué ou est positionné

mise à jour des coordonnées et dimensions des noms d'objet

affichage des noms d'objet (affiche ¨. . .¨ s'il n'y a pas d'objet)

test s'il y a un objet à cet emplacement du sac, si oui affiche l'image, la description et les boutons jeter et utiliser, sinon affiche le texte ¨aucun objet¨

cas où l'objet sélectionné existe dans cet endroit de l'inventaire

affichage de l'image de l'objet ainsi que sa description

regarde yonThrow pour gérer l'affichage du bouton jeter ainsi que sa boîte d'interaction pour jeter un objet

regarde yonUse pour gérer l'affichage du bouton utiliser ainsi que sa boîte d'interaction pour sélectionner un personnage met useOk à 0 si le personnage peut utiliser cet objet, met la variable à 1 sinon

regarde si un personnage a été sélectionné

cas où l'objet sélectionné n'existe pas dans cet endroit de l'inventaire

destruction des textures utilisées pour le menu

destruction des textures utilisées pour le bouton jeter et sa boîte

destruction des textures utilisées pour le bouton utiliser et sa boîte

destruction des textures des noms et descriptions des objets

destruction des textures des noms, pv et pm des personnages

4.21.2.11 ReactEventsInvStuff()

```
void ReactEventsInvStuff (
            int * yonThrow,
            int MouseClick,
            int MouseOver,
            SDL_Renderer * pRenderer,
            int w,
            int h,
            inventaire_t * inv,
            SDL_Rect * destThrowButton,
            team_t * team,
            int numPage,
            int numPagePerso,
            int slot,
            int * eOuStuff )
```

fonction ReactEventsInvStuff qui s'occupe de réagir en fonction des événements quand nous sommes sur le menu d'inventaire pour les équipements/armes

déclaration de i servant aux boucles for et textwidth, textheight pour les dimensions des textes

calcule le nombre de pages nécessaire selon le nombre d'équipements et d'armes

chargement des textures nécessaires au menu

chargement des textures nécessaires au bouton jeter et sa boîte

chargement des textures nécessaires à la boîte ou se situent le personnage et ses slots d'équipements/armes

déclaration des rect correspondant aux coordonnées et dimensions d'affichage du menu

déclaration des rect correspondant aux coordonnées et dimensions d'affichage du perso et des slots d'équiepements/armes

déclaration des rect correspondant aux coordonnées et dimensions d'affichage des boîtes utiliser et jeter ainsi que des personnages pour la boîte d'utilisation

déclaration de la texture et des coordonnés + dimensions des flèches pour changer de page

déclaration de la texture et des coordonnés + dimensions pour les boutons équiper et déséquiper

déclaration et initialisation des textures des noms et descriptions des équipements/armes ainsi que la couleur des texte jaune si équiper, blanc sinon

regarde si la sourie est sur des équipements

mettre à NULL si l'inventaire a atteint sa limite

déclaration de la variable pour les coordonnées et dimensions des noms d'objet

coordonnées et dimensions de la description des objets

coordonnées et destination du bouton jeter

coordonnées et destination des bouton flèches gauche/droite

affichage du menu

calcul des coordonnées/dimensions de la boîte contenant le personnage et affichage

affichage du personnage selon la page

page 1 Golem

page 2 et 3 Toxo/guerrier et xero/mage

destruction de la texture du golem de face

affichage des slots d'équipements

slot 1

afficher un équiement dans le slot s'il est équipé

slot 2

slot 3

slot 4

affichage des slots d'armes

slot 1

slot 2

affichage des équipements/armes et de la zone sur laquelle la souris a cliqué ou est positionnée

affiche seulement s'il y a un équipements/armes à cet emplacement

affichage de la zone autour des noms selon s'il est sélectionné ou non

mise à jour des coordonnées et dimensions des noms des équipements/armes

affichage des noms des équipements/armes

test s'il y a un équipement ou une arme à cet emplacement d'inventaire, si oui affiche l'image, la description et le bouton jeter, et les flèches pour changer de page si possible

affichage de l'image des équipements/armes ainsi que sa description

regarde yonThrow pour gérer l'affichage du bouton jeter ainsi que sa boîte d'interaction pour jeter un objet

cas où l'on a cliqué sur oui

affichage de la flèche de gauche dans l'inventaire puis libère la mémoire

calcul des coordonnées/dimensions et affichage de la flèche droite dans l'inventaire puis libère la mémoire

calcul des coordonnées/dimensions et affichage de la flèche gauche des persos puis libère la mémoire

calcul des coordonnées/dimensions et affichage de la flèche droite des perso puis libère la mémoire

équipe l'équipement ou l'arme séléctionné dans l'inventaire au personnage correspondant au numéro de page des personnages si le bouton équiper est appuyé

chargement->affichage puis libération de la mémoire pour le bouton équiper

déséquipe l'équipement ou l'arme séléctionné dans les slots du personnage si le bouton déséquiper est appuyé

chargement->affichage puis libération de la mémoire pour le bouton déséquiper

cas équipement

cas armes

destruction des textures utilisées pour le menu

destruction des textures utilisées pour le bouton jeter et sa boîte

destruction des textures utilisées pour les slots des équipements/armes

destruction des textures des noms et descriptions des objets

4.21.2.12 ReactEventsOp()

```
void ReactEventsOp (
             int * avancement_quete,
             int MouseOver,
             int KeyIsPressed,
             int * Action,
             SDL_Renderer * pRenderer,
             SDL_Texture * Perso,
             SDL_Rect marcheHori[3][8],
             SDL_Rect * dest,
             int * i,
             int w,
             int h,
             team_t team,
             int biome,
             int choiceShop,
             int * vendeur,
             int colli[125][250],
             SDL_Texture * vendeurDialoTexture[5],
             int * map,
             SDL_Rect * savedest,
             int numperso )
```

fonction ReactEventsOp qui s'occupe de réagir en fonction des événements quand nous sommes sur l'openWorld

déclaration des fonctions ReactEvents pour les sets d'événement possibles selon la scène déclaration des textes pour les pnj

déclaration de la texture de chaque icône

déclaration de la texture des PV et PM de chaque perso

coordonnées et dimensions des PV et PM de chaque persos

coordonnées et dimensions de chaque icône

déclaration des dimensions et des coordonnées de chaque dialogues

déclaration des dimensions et des coordonnées du perso centré

déclaration des dimensions et des coordonnées de chaque dialogue du vendeur

déclaration de la couleur du texte

bloque le personnage si collision est détecté lors du déplacement vers la gauche

bloque le personnage si collision est détecté lors du déplacement vers la droite

bloque le personnage si collision est détecté lors du déplacement vers le haut

bloque le personnage si collision est détecté lors du déplacement vers le bas

cas où le personnage s'arrête vers la droite

cas où le personnage s'arrête vers la gauche

cas où le personnage s'arrête vers le haut

cas où le personnage s'arrête vers le bas

réinitialisation du i pour éviter qu'il dépasse la valeur max d'un int

teste si le personnage est en face d'un pnj vendeur pour faire une interaction une fois la touche f appuyée

teste si le personnage est en face d'un pnj pour faire une interaction une fois la touche f appuyée

teste si le personnage est en face d'un pnj pour faire une interaction une fois la touche f appuyée

test s'il est possible pour le personnage de changer de zone

réinitialise l'action s'il n'y a aucune interaction avec des pnj

chargement des textures selon si la souris est sur l'icône ou non

chargement, affichage et destruction des têtes des persos ainsi que leur nombre PV et de PM

cas où 1 perso est dans l'équipe

chargement de la barre de PV selon le pourcentage de pv du perso

chargement de la barre de Mana selon le pourcentage de Mana du perso, la barre est grisée si ses PM max sont à 0

cas où 2 persos sont dans l'équipe

chargement des têtes des 2 personnages

affichage des têtes des 2 personnage

chargement de la barre de PV selon le pourcentage de pv du perso 1 puis destruction de leur texture

chargement de la barre de PV selon le pourcentage de pv du perso 2

chargement de la barre de Mana selon le pourcentage de Mana du perso 1, la barre est grisée si ses PM max sont à 0

chargement de la barre de Mana selon le pourcentage de Mana du perso 2, la barre est grisée si ses PM max sont à 0

destruction de toutes les textures

cas où 3 persos sont dans l'équipe

chargement des têtes des 3 personnages

affichage des têtes des 3 personnages

chargement de la barre de PV selon le pourcentage de pv du perso 1 puis destruction de leur texture

chargement de la barre de PV selon le pourcentage de pv du perso 2 puis destruction de leur texture

chargement de la barre de PV selon le pourcentage de pv du perso 3

chargement de la barre de Mana selon le pourcentage de Mana du perso 1, la barre est grisée si ses PM max sont à 0

chargement de la barre de Mana selon le pourcentage de Mana du perso 2, la barre est grisée si ses PM max sont à 0

chargement de la barre de Mana selon le pourcentage de Mana du perso 2, la barre est grisée si ses PM max sont à 0

destruction de toutes les textures

affichage des icônes

destruction des textures des icônes

délai d'affichage

4.21.2.13 ReactEventsStats()

```
void ReactEventsStats (
            int numPage,
            int MouseClick,
            int KeyIsPressed,
            int MouseOver,
            SDL_Renderer * pRenderer,
            int w,
            int h,
            team_t * team,
            float Stats[3][8],
            SDL_Rect * destStatAction )
```

fonction ReactEventsInvSac qui s'occupe de réagir en fonction des événements quand nous sommes sur le menu d'inventaire sur le sac

déclaration de i servant aux boucles for et textwidth, textheight pour les dimensions des textes

chaîne de caractère utilisée pour l'affichage des persos avec leur stats

stats stockées temporairement pour vérifier certaines conditions pour pouvoir utiliser le bouton + et -

chargement des textures nécessaires au menu

déclaration des rect correspondant au coordonnées et dimensions d'affichage du menu

déclaration et initialisation des textures des noms et descriptions des objets ainsi que la couleur des textes

déclaration et initialisation des textures des textes des personnages dans la boîte d'utilisation d'objet

déclaration de la variable pour les coordonnées et dimensions des noms d'objet

affichage du menu

création de la texture du nom du perso avec son level

calcul des coordonnées et dimensions du nom du perso avec son lvl

affichage du nom du perso avec son lvl puis libère la mémoire

calcul des coordonnées, dimensions des bouton + et -

chargement de la texture puis affichage du bouton + en on ou off en fonction de si l'on peut utiliser les bouton ou non pour éviter de dépassez le nombre de pt de compétence à utiliser, et libère la mémoire

chargement de la texture puis affichage du bouton - en on ou off en fonction de si l'on peut utiliser les bouton ou non pour éviter de diminuer en-dessous des stats de base, et libère la mémoire

récupère les dernières coordonnées des boutons + et - pour la détection d'events

mise à jour des coordonnées et dimensions des stats

affichage des stats

calcul des coordonnées, dimensions des points de compétence du personnage sur lequel on est

affichage des points de compétence du personnage

calcul des coordonnées, dimensions, charge les textures puis affichage du bouton valider pour confirmer les points de compétence attribués

calcul des coordonnées, dimensions de la flèche gauche pour changer de perso

affiche les flèches pour changer de perso et assombrit le bouton si la souris est dessus
n'affiche pas la flèche de gauche si nous sommes sur le premier perso et n'affiche pas la flèche de droite si nous sommes sur le dernier perso puis libère la mémoire

calcul des coordonnées, dimensions de la flèche droite pour changer de perso puis libère la mémoire

copie des coordonnées des derniers boutons pour les détections d'events

destruction des textures utilisées pour le menu

destruction des textures des noms et descriptions des objets

destruction de toutes les textures des boutons

4.21.2.14 transitionFondu()

```
void transitionFondu (
             SDL_Renderer * renderer,
             int largeur_ecran,
             int hauteur_ecran,
             int duree )
```

Fonction pour réaliser une transition en fondu.

déclaration des fonctions DetectEvents pour les sets d'événement possibles selon la scène Nombre d'images et délai d'affichage

Boucle pour effectuer le fondu

Calculer l'alpha en fonction du progrès de la transition

Affichage du fondu

Mettre à jour l'affichage

Délai d'affichage

4.22 src/fmap.c File Reference

programme gérant la Map

```
#include <string.h>
#include ¨../lib/fmap.h¨
```

Functions

- SDL_Surface * **loadTileset** (const char *filename)
- int **saveSurfaceAsPNG** (SDL_Surface *surface, const char *filename)
- int charger_map (char *CARTE_LIGNE, char *CARTE_TXT, char *nom_carte, int tab[125][250])
- int afficher_map (int coordo_x, int coordo_y, SDL_Renderer *renderer, SDL_Texture *texture_carte)

4.22.1 Detailed Description

programme gérant la Map

Author

François Lépine.

Version

1.8.6

Date

12 Avril 2024.

4.22.2 Function Documentation

4.22.2.1 afficher_map()

```
int afficher_map (
            int coordo_x,
            int coordo_y,
            SDL_Renderer * renderer,
            SDL_Texture * texture_carte )
```

Définir la portion de l'image à afficher

Largeur de la portion à afficher

Hauteur de la portion à afficher

Définir le rectangle de destination

Utiliser la même largeur que la source

Utiliser la même hauteur que la source

4.22.2.2 charger_map()

```
int charger_map (
             char * CARTE_LIGNE,
             char * CARTE_TXT,
             char * nom_carte,
             int tab[125][250] )
```

Ouverture du fichier

Ouvre le fichier tuiles_plaine qui contient les tuiles

Gestion de l'erreur

Variables pour stocker les nombres

Définir la portion de l'image à afficher

Largeur de la portion à afficher

Hauteur de la portion à afficher

Définir le rectangle de destination

Utiliser la même largeur que la source

Utiliser la même hauteur que la source

Récupération de la largeur det de la hauteur

Création de la variable qui va créer le .png contenant l'image de la map

Parcours du fichier contenant la carte en numéro de tuile

Décalage de l'affichage (x et y) sur la map pour pouvoir afficher toute la map

Affichage de la tuile

On va pour chaque tuile enregistrer dans un tableau

Fermeture du fichier

Sauvegarde de la carte en png

4.23 src/fmobs.c File Reference

programme gérant les mobs

```
#include <string.h>
#include ¨../lib/type.h¨
#include ¨../lib/fmobs.h¨
#include ¨../lib/fmap.h¨
```

Functions

- int copie_fichier_mob_txt (char ∗fichier_mob_origine, char ∗fichier_mob_destination)
- int charger_map_mobs (char ∗MOBS_LIGNE, char ∗MOBS_TXT, char ∗MOBS_TXT2, char ∗MAP, int collisions[125][250])
- int detruire_mob (int coordo_x_mob_suppr, int coordo_y_mob_suppr, char ∗MOBS_TXT, int collisions[125][250])

4.23.1 Detailed Description

programme gérant les mobs

programme gérant les fonctions pour les mobs sur la Map

Author

François Lépine.

Version

1.9.5

Date

16 Avril 2024.

4.23.2 Function Documentation

4.23.2.1 charger_map_mobs()

```
int charger_map_mobs (
            char * MOBS_LIGNE,
            char * MOBS_TXT,
            char * MOBS_TXT2,
            char * MAP,
            int collisions[125][250] )
```

Ouvre le fichier mobs_ligne qui contient les créatures

Gestion de l'erreur

Ouvre le fichier de la carte du jeu pour y ajouter les pnj

Gestion de l'erreur

Ouvre le fichier de la carte du jeu pour y ajouter les pnj

Gestion de l'erreur

Ouvre le fichier de la carte du jeu pour y ajouter les pnj

Gestion de l'erreur

Ouvre le fichier de la carte du jeu pour y ajouter les pnj

Gestion de l'erreur

Ouvre le fichier de la carte du jeu pour y ajouter les pnj

Gestion de l'erreur

Ouvre le fichier de la carte du jeu pour y ajouter les pnj

Gestion de l'erreur

Variables pour stocker les nombres

Définir la portion de l'image à afficher

Largeur de la portion à afficher

Les monstres ont un sprite contenant 6 animations

Hauteur de la portion à afficher

Définir le rectangle de destination

Utiliser la même largeur que la source

Utiliser la même hauteur que la source

Ouverture du fichier

Récupération de la largeur det de la hauteur

Parcours du fichier contenant la carte en numéro de tuile

Dans le tableau de collisions il y aura 3 créatures, chacune d'elles aura une seule et même configuration

Affichage de la tuile

Affichage de la tuile

Affichage de la tuile

Affichage de la tuile

Affichage de la tuile

Affichage de la tuile

Fermeture du fichier

Sauvegarde de la carte en png

Récupération de la largeur det de la hauteur

Parcours du fichier contenant la carte en numéro de tuile

Dans le tableau de collisions il y aura 3 créatures, chacune d'elles aura une seule et même configuration

Affichage de la tuile

Affichage de la tuile

Affichage de la tuile

Affichage de la tuile

Affichage de la tuile

Affichage de la tuile

Fermeture du fichier

Sauvegarde de la carte en png

4.23.2.2 copie_fichier_mob_txt()

```
int copie_fichier_mob_txt (
            char * fichier_mob_origine,
            char * fichier_mob_destination )
```

Ouvrir le fichier destination en mode lecture

Ouvrir le fichier destination en mode écriture

Copier le contenu du fichier source vers le fichier destination

Fermer les fichiers

4.23.2.3 detruire_mob()

```
int detruire_mob (
            int coordo_x_mob_suppr,
            int coordo_y_mob_suppr,
            char * MOBS_TXT,
            int collisions[125][250] )
```

Ouverture du fichier

Ouvrir un fichier temporaire en mode écriture

Lire chaque ligne du fichier source

Extraire les deux derniers chiffres de la ligne

Si les deux derniers chiffres ne correspondent pas à ceux spécifiés, écrire la ligne dans le fichier temporaire

Fermer les fichiers

Supprimer le fichier source

Renommer le fichier temporaire pour qu'il devienne le nouveau fichier source

4.24 src/fpnj.c File Reference

programme gérant les pnj sur la map

```
#include <string.h>
#include ¨../lib/type.h¨
#include ¨../lib/fpnj.h¨
#include ¨../lib/fmap.h¨
```

Functions

- int charger_map_pnj (char ∗PNJ_LIGNE, char ∗PNJ_TXT, char ∗MAP, char ∗nom_carte, int collisions[125][250])

4.24.1 Detailed Description

programme gérant les pnj sur la map

Author

François Lépine.

Version

1.8.6

Date

12 Avril 2024.

4.24.2 Function Documentation

4.24.2.1 charger_map_pnj()

```
int charger_map_pnj (
            char * PNJ_LIGNE,
            char * PNJ_TXT,
            char * MAP,
            char * nom_carte,
            int collisions[125][250] )
```

Ouverture du fichier

Ouvre le fichier pnj_ligne qui contient les pnj

Gestion de l'erreur

Ouvre le fichier de la carte du jeu pour y ajouter les pnj

Gestion de l'erreur

Variables pour stocker les nombres

Définir la portion de l'image à afficher

Largeur de la portion à afficher

Hauteur de la portion à afficher

Définir le rectangle de destination

Utiliser la même largeur que la source

Utiliser la même hauteur que la source

Récupération de la largeur det de la hauteur

Parcours du fichier contenant la carte en numéro de tuile

Affichage de la tuile

Fermeture du fichier

Sauvegarde de la carte en png

4.25 src/GolemAdventure.c File Reference

programme principal gérant tout le jeu.

```
#include <stdio.h>
#include <stdlib.h>
#include ¨../lib/EventOp.h¨
#include ¨../lib/utils.h¨
#include ¨../lib/init_menu.h¨
#include ¨../lib/combat.h¨
```

Functions

- int main (int argc, char ∗∗argv)

4.25.1 Detailed Description

programme principal gérant tout le jeu.

Author

Cognard Luka, Bonga Warrick, François Lépine, Lenny Borry.

Version

1.9.5

Date

16 Avril 2024.

4.25.2 Function Documentation

4.25.2.1 main()

```
int main (
            int argc,
            char ** argv )
```

initialisation de x pour les coordonnées src et i pour les boucles
ainsi que de w et h pour récupérer la longueur et largeur de la fenêtre

Initialisation de SDL pour les VIDEO, EVENTS et AUDIO

Création de la fenêtre

Initialisation de SDL_IMAGE

Initialisation de SDL_TTF

Initialisation de SDL_Mixer

Création d'un SDL_Renderer utilisant l'accélération matérielle

chargement des textures du perso, etc ...

déclaration des destinations du perso, map, block et pnj ainsi que des variables pour la musique

création et initialisation d'un tableau sélectionnant tous les sprites de l'animation de marche

déclaration de quitOp et quitMenu étant les variables booléennes qui mettent fin respectivement à la boucle de l'open world et des menus déclaration keyPressed pour gérer les touches déclaration Action pour les interactions déclaration MouseHover, Mouseclick pour gérer la souris déclaration yonThrow gérant la boîte d'interaction pour jeter un objet déclaration yonUse gérant la boîte d'interaction pour utiliser un objet déclaration tableau useOk étant à 1 si l'objet n'est pas utilisable sur un personnage, à 0 sinon

déclaration de la scène où est le jeu

création de tous les objets du jeu

création de tous les équipements du jeu en 3 fois, pour éviter d'avoir le même équipement sur plusieurs perso

création de tous les personnages possibles

déclaration de l'équipe en jeu pour tester

déclaration et initialisation de l'inventaire

ajoute de l'équipement dans l'inventaire pour tester

création de toutes les armes

déclaration de la destination et des dimensions puis chargement de la texture pour la cinématique du début

boucle entière du jeu

boucle menu

boucle open world

récupération de la longueur et de la largeur de la fenêtre

Regarde sur quelle scène nous sommes pour gérer les événements et l'affichage

Open World

activation de la musique

détection des événements

réaction aux événements

nettoyage de l'écran et affichage de la map

cas de la map open world

cas de la map bibliothèque

Menu Pause

destruction des textures des personnages et des map puis réinitialistion de toutes les variables

réaction aux événements

Menu Inventaire sac

met la touche de déplacement utilisée à son état d'arrêt pour afficher le personnage dans la bonne direction

nettoyage de l'écran et affichage de la map

Menu Inventaire des stats

met la touche de déplacement utilisée à son état d'arrêt pour afficher le personnage dans la bonne direction

garde le numéro de page entre le nombre de persos et la page 1

nettoyage de l'écran et affichage de la map

Menu Inventaire des équipements

met la touche de déplacement utilisée à son état d'arrêt pour afficher le personnage dans la bonne direction

garde le numéro de page entre le nombre max de pages calculé et la page 1

garde le numéro de pages entre le nombre de perso et la page 1

empêche le joueur de sélectionner une zone d'équipement ou d'arme où il n'y a rien

Menu du magasin pour acheter de l'équipement, des armes et des objets

nettoyage de l'écran et affichage de la map

Menu de vente pour vendre notre équipement et nos armes

nettoyage de l'écran et affichage de la map

met la touche de déplacement utilisée à son état d'arrêt pour afficher le personnage dans la bonne direction

Mise à jour de l'écran

réinitialisation de l'inventaire

Libération de la mémoire associée aux Musiques

Libération de la mémoire associée aux textures

Libération de la mémoire du SDL_Renderer

fermeture de toutes les extensions SDL, IMG, TTF et AUDIO

4.26 src/lib_menu.c File Reference

programme gérant les Menus.

```
#include ¨../lib/init_menu.h¨
```

Functions

- void **readSaveName** (char name[20], FILE ∗save)
  Lecture du nom d'une sauvegarde.
- int menuPrincipal (SDL_Renderer ∗renderer)
  Menu principal.
- int menuPlay (SDL_Renderer ∗renderer)
  Menu PLAY.
- int menuSettings (SDL_Renderer ∗renderer)
  Menu SETTINGS.
- int menuCredits (SDL_Renderer ∗renderer)
  Menu CREDITS.
- int menuPause (SDL_Renderer ∗renderer, scene_t ∗scene, SDL_Texture ∗map, SDL_Rect dest)
  Menu PAUSE.

4.26.1 Detailed Description

programme gérant les Menus.

Author

Warrick Bonga.

Version

1.8.6

Date

12 Avril 2024.

4.26.2 Function Documentation

4.26.2.1 menuCredits()

```
int menuCredits (
            SDL_Renderer * renderer )
```

Menu CREDITS.

Chargement de l'image du bouton retour par defaut

Chargement de l'image du bouton retour au survol

Chargement de l'image du bouton retour lors du clic

Effacement de l'écran

Affichage de l'image appropriée

Affichage sur l'écran

Libération des ressources

4.26.2.2 menuPause()

```
int menuPause (
            SDL_Renderer * renderer,
            scene_t * scene,
            SDL_Texture * map,
            SDL_Rect dest )
```

Menu PAUSE.

Chargement de l'image de sauvegarde par defaut

Chargement de l'image du bouton quitter par defaut

Chargement de l'image Options par defaut

Chargement de l'image du bouton retour par défaut

Chargement de l'image de sauvegarde au survol

Chargement de l'image du bouton quitter au survol

Chargement de l'image Options au survol

Chargement de l'image du bouton retour au survol

Chargement de l'image de sauvegarde lors du clic

Chargement de l'image du bouton quitter lors du clic

Chargement de l'image Options lors du clic

Chargement de l'image du bouton retour lors du clic

condition d'arrêt : sélection d'un choix du menu de pause OU appui sur [échap]

Effacement de l'écran

Affichage de l'image appropriée

Affichage sur l'écran

Libération des ressources

4.26.2.3 menuPlay()

```
int menuPlay (
            SDL_Renderer * renderer )
```

Menu PLAY.

Chargement du fond du menu play

Chargement de l'image de la sauvegarde 1 par défaut

Chargement de l'image de la sauvegarde 2 par défaut

Chargement de l'image de la sauvegarde 3 par défaut

Chargement de l'image de la sauvegarde 1 au survol

Chargement de l'image de la sauvegarde 2 au survol

Chargement de l'image de la sauvegarde 3 au survol

Chargement de l'image de la sauvegarde 1 au clic

Chargement de l'image de la sauvegarde 2 au clic

Chargement de l'image de la sauvegarde 3 au clic

Chargement de l'image de l'option lancement par défaut

Chargement de l'image de l'option écrasement par défaut

Chargement de l'image de l'option retour par défaut

Chargement de l'image de l'option lancement au survol

Chargement de l'image de l'option écrasement au survol

Chargement de l'image de l'option retour au survol

Chargement de l'image de l'option lancement au clic

Chargement de l'image de l'option écrasement au clic

Chargement de l'image de l'option retour au clic

condition d'arrêt : chargement d'une sauvegarde (vide ou commencée) OU retour à l'écran-titre

Effacement de l'écran

Affichage de l'image appropriée

Affichage sur l'écran

Libération des ressources

4.26.2.4 menuPrincipal()

```
int menuPrincipal (
            SDL_Renderer * renderer )
```

Menu principal.

Fonctions des menus. Chargement de l'image Play par defaut

Chargement de l'image Settings par defaut

Chargement de l'image Credits par defaut

Chargement de l'image Play au survol

Chargement de l'image Settings au survol

Chargement de l'image Credits au survol

Chargement de l'image Play lors du clic

Chargement de l'image Settings lors du clic

Chargement de l'image Credits lors du clic

condition d'arrêt : choix d'un menu

Effacement de l'écran

Affichage de l'image appropriée

Affichage sur l'écran

Libération des ressources

4.26.2.5 menuSettings()

```
int menuSettings (
            SDL_Renderer * renderer )
```

Menu SETTINGS.

Chargement de l'image du bouton retour par defaut

Chargement de l'image du bouton retour au survol

Chargement de l'image du bouton retour lors du clic

Effacement de l'écran

Affichage de l'image appropriée

Affichage sur l'écran

Libération des ressources

4.27 src/test_unit.c File Reference

programme faisant les tests unitaires.

```
#include <CUnit/CUnit.h>
#include <CUnit/Basic.h>
#include <stdio.h>
#include ¨../lib/EventOp.h¨
#include ¨../lib/texture.h¨
#include ¨../lib/type.h¨
#include ¨../lib/init_menu.h¨
```

Functions

- void **test_creer_texture** ()
  test la fonction de création de texture
- void **test_creer_texture_texte** ()
- void **test_init_team** ()
  test la fonction d'initialisation de la team
- void **test_init_inv** ()
  test la fonction d'initialisation de l'inventaire
- int **main** ()
  programme principal effectuant tous les tests

4.27.1 Detailed Description

programme faisant les tests unitaires.

Author

Cognard Luka.

Version

1.9.5

Date

16 Avril 2024.

4.28 src/texture.c File Reference

programme gérant les fonctions de chargement de texture.

```
#include ¨../lib/texture.h¨
```

Functions

- SDL_Texture * loadTexture (const char *chemin, SDL_Renderer *renderer)

  fonction loadTexture qui s'occupera de charger l'image avec une surface puis d'utiliser cette surface pour créer une texture la fonction renvoie NULL s'il y a une erreur, renvoie la texture créée sinon
- SDL_Texture * loadTextureFont (const char *chemin, SDL_Renderer *renderer, const char *text, int h, SDL_Color color)

  fonction loadTextureFont qui s'occupera de charger le texte avec une surface puis d'utiliser cette surface pour créer une texture la fonction renvoie NULL si il y a une erreur, renvoie la texture créée sinon

4.28.1 Detailed Description

programme gérant les fonctions de chargement de texture.

Author

Luka Cognard.

Version

1.8.6

Date

12 Avril 2024.

4.28.2 Function Documentation

4.28.2.1 loadTexture()

```
SDL_Texture * loadTexture (
            const char * chemin,
            SDL_Renderer * renderer )
```

fonction loadTexture qui s'occupera de charger l'image avec une surface puis d'utiliser cette surface pour créer une texture la fonction renvoie NULL s'il y a une erreur, renvoie la texture créée sinon

déclaration de la fonction loadTexture qui prend en paramètre le chemin de l'image et le renderer déclaration de la fonction loadTextureFont qui prend en paramètre le chemin de la police d'écriture, le renderer, le texte à afficher, la taille de police et la couleur du texte crée la surface de l'image, renvoie NULL avec une erreur si ça n'a pas marché

crée la texture de l'image à partir de ça surface puis libère la surface inutiliser renvoie NULL avec une erreur si la texture n'est pas créée

4.28.2.2 loadTextureFont()

```
SDL_Texture * loadTextureFont (
            const char * chemin,
            SDL_Renderer * renderer,
            const char * text,
            int h,
            SDL_Color color )
```

fonction loadTextureFont qui s'occupera de charger le texte avec une surface puis d'utiliser cette surface pour créer une texture la fonction renvoie NULL si il y a une erreur, renvoie la texture créée sinon

charge la police d'écriture via le chemin entré en paramètre renvoie NULL avec une erreur si ça n'a pas marché

crée la surface du texte via le texte entré en paramètre libère la police chargée inutilisée renvoie NULL avec une erreur si ça n'a pas marché

crée la texture du texte à partir de sa surface puis libère la surface inutilisée renvoie NULL avec une erreur si la texture n'est pas créée

4.29 src/type.c File Reference

programme gérant les types

```
#include ¨../lib/type.h¨
#include ¨../lib/texture.h¨
#include <stdio.h>
#include <string.h>
```

Functions

- attaque_t **creer_attaque** (char nom[TAILLE_NOM], type_att_t type_att, char desc[100], effet_t effet, int qte↩ _effet, int duree_effet, int nb_cible, int precision)

  declaration de la fonction creer_attaque
- mob_t ∗ creer_mob (char ∗nom, SDL_Renderer ∗ecran, int lvl)

  déclaration des fonctions en rapport avec les mobs
- void afficher_arme (arme_t ∗arme)

  fonction qui affiche les armes
- void **creer_all_arme** (arme_t armes[ALL_ARMES][3], SDL_Renderer ∗ecran)

  fonction qui crée toutes les armes
- void **creer_all_att_arme** (arme_t armes[ALL_ARMES][3])

  fonction qui crée les attaques de chaque arme
- void equiper_arme (arme_t ∗arme, perso_t ∗perso)

  fonction qui équipe un équipement d'un personnage
- int **desequiper_arme** (arme_t ∗arme, perso_t ∗perso)

  fonction qui déséquipe un équipement d'un personnage
- int ajouter_arme (arme_t allArmes[ALL_ARMES][3], inventaire_t ∗inv, int indArm)

  fonction qui ajoute une arme dans l'inventaire
- int **jeter_arme** (inventaire_t ∗Inv, arme_t ∗arme)

  fonction qui supprime une arme de l'inventaire

- int creer_perso (perso_t *perso, char *nom, char *classe, SDL_Renderer *ecran, float *statPerso)

  fonction créer un perso, renvoie NULL si la création n'a pas fonctionné
- void **lvl_up** (perso_t *perso, float *statPerso)

  met a jour le perso quand il a assez d'xp pour lvl up
- void creer_all_objet (objet_t allObj[9], SDL_Renderer *ecran)

  fonction qui va créer un objet_t avec sa texture entrée en paramètre
- void **utiliser_obj** (inventaire_t *Inv, perso_t *perso, int iObj)

  fonction qui va utiliser l'objet et le retirer de l'inventaire
- int **jeter_obj** (inventaire_t *Inv, int iObj)

  fonction supprime un objet de l'inventaire
- int **ajouter_obj** (inventaire_t *Inv, objet_t *allObj, int iObj)

  fonction ajoute un objet dans l'inventaire
- void init_and_reinit_inv (inventaire_t *inv)

  fonction qui initialise l'inventaire et permet de le réinitialiser
- void **aff_perso** (perso_t perso)

  affichage d'un perso pour des tests ou autre (effaçable une fois inutile)
- void creer_all_sprite_equipement (SDL_Texture *allSprite[ALL_EQUIPEMENTS], SDL_Renderer *ecran)

  charge toutes les textures de chaque équipement
- void creer_all_equipement (equipement_t allStuff[ALL_EQUIPEMENTS][3], SDL_Texture *allSprite[ALL_↩ EQUIPEMENTS])

  créer le tableau avec tous les équipements en 3 fois pour éviter qu'un personnage soit équipé d'exactement le même équipement
- int ajouter_stuff (equipement_t allStuff[ALL_EQUIPEMENTS][3], inventaire_t *inv, int iStuff)

  fonction qui ajoute un équipement dans l'inventaire
- void equiper_stuff (equipement_t *equipement, perso_t *perso)

  fonction qui équipe un équipement d'un personnage
- int **desequiper_stuff** (int iStuff, perso_t *perso)

  fonction qui déséquipe un équipement d'un personnage
- int jeter_stuff (inventaire_t *Inv, int iStuff)

  fonction qui supprime un équipement de l'inventaire
- int **join_team** (team_t *team, perso_t *allPerso)

  déclaration des fonctions liées à l'équipe du joueur
- void **init_reinit__team** (team_t *team)

4.29.1 Detailed Description

programme gérant les types

Author

Luka Cognard, Lenny Borry.

Version

1.9.5

Date

12 Avril 2024.

4.29.2 Function Documentation

4.29.2.1 afficher_arme()

```
void afficher_arme (
            arme_t * arme )
```

fonction qui affiche les armes

declaration des fonctions en rapport avec les armes et les attaques

4.29.2.2 ajouter_arme()

```
int ajouter_arme (
            arme_t allArmes[ALL_ARMES][3],
            inventaire_t * inv,
            int indArm )
```

fonction qui ajoute une arme dans l'inventaire

regarde si l'inventaire n'est pas plein

regarde combien de fois il possède cet équipement, impossible d'ajouter l'objet s'il l'a 3 fois

ajoute l'équipement avec le numId qui n'est pas présent

renvoie 2 s'il a déja 3 fois l'équipement voulu

renvoie 1 si l'inventaire est plein

4.29.2.3 ajouter_stuff()

```
int ajouter_stuff (
            equipement_t allStuff[ALL_EQUIPEMENTS][3],
            inventaire_t * inv,
            int iStuff )
```

fonction qui ajoute un équipement dans l'inventaire

regarde si l'inventaire n'est pas plein

regarde combien de fois il possède cet équipement, impossible d'ajouter l'objet s'il l'a 3 fois

ajoute l'équipement avec le numId qui n'est pas présent

renvoie 2 s'il a déja 3 fois l'équipement voulu

renvoie 1 si l'inventaire est plein

4.29.2.4 creer_all_equipement()

```
void creer_all_equipement (
             equipement_t allStuff[ALL_EQUIPEMENTS][3],
             SDL_Texture * allSprite[ALL_EQUIPEMENTS] )
```

créer le tableau avec tous les équipements en 3 fois pour éviter qu'un personnage soit équipé d'exactement le même équipement

créer tout les casques, plastrons et bottes par biome

créer les talismans

4.29.2.5 creer_all_objet()

```
void creer_all_objet (
             objet_t allObj[9],
             SDL_Renderer * ecran )
```

fonction qui va créer un objet_t avec sa texture entrée en paramètre

toutes les valeurs des différents objets dans des tableaux

création des objets

4.29.2.6 creer_all_sprite_equipement()

```
void creer_all_sprite_equipement (
             SDL_Texture * allSprite[ALL_EQUIPEMENTS],
             SDL_Renderer * ecran )
```

charge toutes les textures de chaque équipement

déclaration des fonctions liées aux équipements

4.29.2.7 creer_mob()

```
mob_t * creer_mob (
             char * nom,
             SDL_Renderer * ecran,
             int lvl )
```

déclaration des fonctions en rapport avec les mobs

change les stats de base selon la classe du perso

4.29.2.8 creer_perso()

```
int creer_perso (
            perso_t * perso,
            char * nom,
            char * classe,
            SDL_Renderer * ecran,
            float * statPerso )
```

fonction créer un perso, renvoie NULL si la création n'a pas fonctionné

déclaration des fonctions liées aux personnages (équiper un équipement/arme ou utiliser un objet, lvl up) change les stats de base selon la classe du perso

initialise les tableaux de pointeurs

copie les stats des persos

4.29.2.9 equiper_arme()

```
void equiper_arme (
            arme_t * arme,
            perso_t * perso )
```

fonction qui équipe un équipement d'un personnage

si l'arme est équipée sur un personnage, on la déséquipe

4.29.2.10 equiper_stuff()

```
void equiper_stuff (
            equipement_t * equipement,
            perso_t * perso )
```

fonction qui équipe un équipement d'un personnage

le déséquipe si l'équipement est équipé sur un personnage

le déséquipe de cet objet si un équipement du même type est déjà équipé

4.29.2.11 init_and_reinit_inv()

```
void init_and_reinit_inv (
            inventaire_t * inv )
```

fonction qui initialise l'inventaire et permet de le réinitialiser

déclaration des fonction liées à l'inventaire avec le sac d'objets, les équipements et les armes

4.29.2.12 jeter_stuff()

```
int jeter_stuff (
            inventaire_t * Inv,
            int iStuff )
```

fonction qui supprime un équipement de l'inventaire

si l'équipement est équipé sur un personnage le déséquipe avant de jeter l'équipement

4.30 src/utils.c File Reference

programme gérant le menu pour rentrer un pseudo, ainsi que les sauvegardes.

```
#include ¨../lib/utils.h¨
```

Functions

- void **entrerLettre** (SDL_Event event, char ∗name, int ∗indice, SDL_Color color, int ∗quit)

  Fonction de détection de chaque touche utile du clavier pour entrer un nom : condition qui permet de vérifier si la longueur est bien inférieure à la taille du nom maximale (TAILLE_NOM - 1) pour pouvoir insérer un caractère condition qui permet de vérifier si la longueur est bien supérieure à 0 pour pouvoir supprimer un caractère condition qui permet de vérifier si la longueur est bien supérieure à 1 pour pouvoir valider le nom si et seulement si l'indice pointe sur le début de la chaîne, la lettre sera en majuscule.
- char ∗ entrerNom (SDL_Renderer ∗renderer)

  Fonction d'entrée du nom.
- void sauvegarder (int nb_save, int idMap, int x, int y, team_t team, inventaire_t inv, int lorePos, objet_t allPot[ALL_OBJETS], equipement_t allStuff[ALL_EQUIPEMENTS][3], arme_t allArmes[ALL_ARMES][3], SDL_Renderer ∗renderer, SDL_Texture ∗Map, SDL_Rect dest)

  Fonction de sauvegarde de la progression.
- void chargerSave (int nb_save, int ∗idMap, int ∗x, int ∗y, team_t ∗team, inventaire_t ∗inv, int ∗lorePos, objet_t allPot[ALL_OBJETS], equipement_t allStuff[ALL_EQUIPEMENTS][3], perso_t allPerso[NB_TEAM_↩ PERSOS], arme_t allArmes[ALL_ARMES][3], float stats[NB_TEAM_PERSOS][8], SDL_Renderer ∗renderer)

  Fonction de chargement de la sauvegarde.
- void nouvelleSave (int nb_save, char ∗nom)

  Fonction d'initialisation de la sauvegarde.

4.30.1 Detailed Description

programme gérant le menu pour rentrer un pseudo, ainsi que les sauvegardes.

Author

Warrick Bonga.

Version

1.9.8

Date

12 Avril 2024.

4.30.2 Function Documentation

4.30.2.1 chargerSave()

```
void chargerSave (
            int nb_save,
            int * idMap,
            int * x,
            int * y,
            team_t * team,
            inventaire_t * inv,
            int * lorePos,
            objet_t allPot[ALL_OBJETS],
            equipement_t allStuff[ALL_EQUIPEMENTS][3],
            perso_t allPerso[NB_TEAM_PERSOS],
            arme_t allArmes[ALL_ARMES][3],
            float stats[NB_TEAM_PERSOS][8],
            SDL_Renderer * renderer )
```

Fonction de chargement de la sauvegarde.

Ouverture du fichier contenant la sauvegarde

Déclaration des indices, de l'élément courant et du caractère courant lors de la lecture du fichier

Déclaration des variables provisoires permettant la lecture des statistiques

Chargement de la dernière carte du monde sauvegardée ainsi que la position du personnage dessus

Chargement des personnages et de leurs statistiques

Chargement des objets de l'inventaire (potions)

Chargement de l'équipement de l'inventaire (armure) et de la bourse (montant de la monnaie du jeu)

Chargement des armes de l'inventaire

Chargement de l'avancement dans l'histoire du jeu

Fermeture du fichier contenant la sauvegarde

4.30.2.2 entrerNom()

```
char * entrerNom (
            SDL_Renderer * renderer )
```

Fonction d'entrée du nom.

Déclaration des variables d'event et d'arrêt de la boucle

Déclaration des chaînes de caractères de la fenêtre (en-tête et règles d'entrée du nom)

Déclaration de la couleur blanche avec le type personalisé de SDL

Création des textures des textes déclarés plus haut

Création et allocation des zones des textures + copie dans renderer

Déclaration du nom et de l'indice d'écriture dans le nom

Premier bourrage de la chaîne du nom pour éviter un affichage de caractères aléatoires

Boucle d'entrée du nom

L'entrée de lettres s'activera au moment de l'appui sur une des touches disponibles (cf. fonction entrerLettre)

Pour éviter une erreur lorsque la largeur de la texture vaut 0, le premier caractère du nom vaut '_' lorsque celui-ci est vide

Écriture du nom sur l'écran

Rafraîchissement de l'écran

Libération en mémoire des textures

Appuyer sur échap retourne le nom ¨Golem¨ par défaut

Bourrage de la place allouée non utilisée

Renvoi de l'adresse du nom

4.30.2.3 nouvelleSave()

```
void nouvelleSave (
            int nb_save,
            char * nom )
```

Fonction d'initialisation de la sauvegarde.

Création du dossier contenant les sauvegardes

Ouverture du fichier d'initialisation de la sauvegarde

Initialisation de la première carte du monde ainsi que la position du personnage au départ

Initialisation du personnage de départ et de ses statistiques

Intialisation de l'emplacement des objets de l'inventaire (potions)

Intialisation de l'emplacement de l'équipement de l'inventaire (armure) et de la bourse (montant de la monnaie du jeu)

Initialisation de l'emplacement des armes de l'inventaire

Initialisation de la valeur représentant l'avancement dans l'histoire du jeu

Fermeture du fichier d'initialisation de la sauvegarde

4.30.2.4 sauvegarder()

```
void sauvegarder (
             int nb_save,
             int idMap,
             int x,
             int y,
             team_t team,
             inventaire_t inv,
             int lorePos,
             objet_t allPot[ALL_OBJETS],
             equipement_t allStuff[ALL_EQUIPEMENTS][3],
             arme_t allArmes[ALL_ARMES][3],
             SDL_Renderer * renderer,
             SDL_Texture * Map,
             SDL_Rect dest )
```

Fonction de sauvegarde de la progression.

Déclaration de l'image pour la fenêtre de la sauvegarde

Déclaration des images textuelles pendant et après la sauvegarde

Ouverture du fichier contenant la sauvegarde

Déclaration des indices

Affichage du texte pendant la sauvegarde

Sauvegarde de la dernière carte du monde chargée ainsi que la position du personnage dessus

Sauvegarde des personnages et de leurs statistiques

Sauvegarde des objets de l'inventaire (potions)

Sauvegarde de l'équipement de l'inventaire (armure) et de la bourse (montant de la monnaie du jeu)

Sauvegarde des armes de l'inventaire

Sauvegarde de l'avancement dans l'histoire du jeu

Affichage du texte après la sauvegarde

Affichage du texte jusqu'à qu'un clic de souris soit détecté

Destruction des textures

Fermeture du fichier contenant la sauvegarde

Index