Java_Project

Generated by Doxygen 1.9.4

Chapter 1

aFaire

Liste des choses à faire :

- Changement de l'appel des directions ("up", "down", ...) par un enum (moins couteux en espace mémoire)
- Optimisation de performance pour le rendu des tiles à refaire & revoir (faire un rendu autour du joueur de juste un morceau de la map)
- Vérifier d'où vient le problème de vitesse dans un sens plutôt que dans l'autre

Questions à poser :

- Est-ce que sortir le fichier main.java du package main améliorerait la structure du code ?
- Est-ce que mettre l'entièreté de la bibliothèque est recommandé ou non ?
- Est-ce que dans les notions à implémenter, le tableau est considéré comme à taille variable ou non ?

Chapter 2

Projet Objet en Java

Projet POO java 4A

Lien vers le formulaire des règles : https://docs.google.com/document/d/1psWO6b7h8↩
CZb7-FEe0dCSoRnp4pSnqK-fcdwGJWe_0M/edit?usp=sharing

2.1 Avancé du projet

2.1.0.1 06/10/23 :

- Dégrossissement du projet et choix de la conception du jeu
- Établissement des règles globales
- Création du doc de résumé : https://docs.google.com/document/d/1psWO6b7h8↩
CZb7-FEe0dCSoRnp4pSnqK-fcdwGJWe_0M/edit?usp=sharing

2.1.0.2 13/10/23 :

- Commencé à coder le jeu : affichage de la fenêtre

2.1.0.3 16/10/23 :

- Création du Game Loop

2.1.0.4 18/10/23 :

- Création de l'UML, lien du fichier : https://lucid.app/lucidchart/d732d001-fe42-4d1a-81b0-fa0eec
_loc=-1159%2C-232%2C2219%2C948%2C0_0&invitationId=inv_12128bfc-8225-4790-a92b-9c527
- Class crées : main, entity, tiles
- Finalisation du doc : Définition de l'univers et des règles

2.1.0.5 29/10/23 :

- Refonte de l'UML et modification régulière de celui-ci.

2.1.0.6 14/11/23 :

- Finition de l'UML et organisation des tâches pour les contributeurs
- Création de map de debug et de test

2.1.0.7 15/11/23 :

- Implémentation du patron de conception : singleton (romu)

2.1.0.8 19/11/23 :

- Refonte du système de mapping du TileManager
- Ajout d'animations de plusieurs décors (sans interractions)

Chapter 3

Namespace Index

3.1 Package List

Here are the packages with brief descriptions (if available):

Chapter 4

Hierarchical Index

4.1 Class Hierarchy

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

Chapter 5

Class Index

5.1 Class List

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

Chapter 6

File Index

6.1 File List

Here is a list of all files with brief descriptions:

Chapter 7

Namespace Documentation

7.1 Package CreationCombatScene

Packages

- package entity
- package game

Classes

- class Main

7.2 Package CreationCombatScene.entity

Classes

- class PlayerTest

7.3 Package CreationCombatScene.game

Classes

- class Window

7.4 Package entity

Packages

- package props

Classes

- class Character

  Represents an abstract character entity with position, hitbox, stats, and animations.
- class Enemy

  Represents an enemy entity in the game.
- class Entity

  Represents an abstract entity with position, hitbox, and animations.
- class EntitySetter

  Responsible for initializing and setting up objects (entities) in the game world.
- class Player

  Represents the player entity in the game.

7.5 Package entity.props

Classes

- class OBJ_Chest

  Represents a chest object.
- class OBJ_Door

  Represents a door object in the game.
- class OBJ_Key

  Represents a key object in the game.
- class Props

  Represents in-game props with properties such as image, name, and position.

7.6 Package game

Classes

- class Const
- class FightScene

  Represents the scene during a fight between a player and an enemy.
- class Scene

  Represents an abstract scene in the game.
- class Window

  Represents the window that displays the game.
- class World

  Represents the game world and manages its entities.

7.7 Package item

Packages

- package armor
- package potion
- package weapon

Classes

- class Generator
- class Item

7.8 Package item.armor

Classes

- class Armor
- class Body
- class Foot
- class Head
- class Legs

7.9 Package item.potion

Classes

- class HealthPotion
- class ManaPotion
- class Potion
- class SpeedPotion

7.10 Package item.weapon

Classes

- class Bow
- class Staff
- class Sword
- class Weapon

7.11 Package main

Classes

- class KeyHandler

  Handles keyboard input using the singleton pattern.
- class Main

  Contains the main method to start the 2D Adventure game.

7.12 Package tiles

Classes

- class Tile

  Represents a tile in the game world.
- class TileManager

  Manages tiles in the game world.

7.13 Package UI

Classes

- class ChoiceButton
- class HUD
- class Textbox

Chapter 8

Class Documentation

8.1 item.armor.Armor Class Reference

Inheritance diagram for item.armor.Armor:

Collaboration diagram for item.armor.Armor:

8.1.1 Detailed Description

Definition at line 5 of file Armor.java.

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

- src/item/armor/Armor.java

8.2 item.armor.Body Class Reference

Inheritance diagram for item.armor.Body:

Collaboration diagram for item.armor.Body:

Public Member Functions

- Body ()

8.2.1 Detailed Description

Definition at line 3 of file Body.java.

8.2.2 Constructor & Destructor Documentation

8.2.2.1 Body()

```
item.armor.Body.Body ( )
```

Definition at line 5 of file Body.java.

```
00005                 {
00006         System.out.println("Armure body");
00007     }
```

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

- src/item/armor/Body.java

8.3 item.weapon.Bow Class Reference

Inheritance diagram for item.weapon.Bow:

Collaboration diagram for item.weapon.Bow:

Public Member Functions

- Bow ()

8.3.1 Detailed Description

Definition at line 3 of file Bow.java.

8.3.2 Constructor & Destructor Documentation

8.3.2.1 Bow()

```
item.weapon.Bow.Bow ( )
```

Definition at line 5 of file Bow.java.

```
00005                 {
00006         System.out.println("Bow");
00007     }
```

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

- src/item/weapon/Bow.java

8.4 entity.Character Class Reference

Represents an abstract character entity with position, hitbox, stats, and animations.

Inheritance diagram for entity.Character:

Collaboration diagram for entity.Character:

Public Member Functions

- Character (String entityName, int x, int y, int dirX, int dirY, int speed, String facing, int _spriteCntMax, int spriteSpeed)

  Constructor for the Character class.
- void update (Scene scene, double dt)

  Updates the character entity based on the current scene and time elapsed.
- void drawInWorld (Graphics2D g2, int screenX, int screenY)

  Draws the character in the world scene.
- void drawInFight (Graphics2D g2, int screenX, int screenY)

  Draws the character in the fight scene.

Public Attributes

- int speed
- int dirX
- int dirY
- String facing

Protected Member Functions

- void move (World world, int speed, double dt)

  Moves the character in the world based on its direction, speed, and the elapsed time.
- void accelerate (int maxSpeed, int factor, double dt)

  Accelerates the character's speed up to the specified maximum speed.
- void decelerate (int factor, double dt)

  Decelerates the character's speed.
- void loadTextures (String name)

  Loads character textures based on the specified name.
- void playerInterraction (Player player)

Package Attributes

- int health
- int mana
- int agility
- int strength
- int defense
- int initiative
- int hasKey = 0

Private Member Functions

- void checkTileCollision (TileManager tileManager)

  Checks for collision with nearby tiles using the character's hitbox.

Private Attributes

- BufferedImage[ ] _idle_up
- BufferedImage[ ] _idle_down
- BufferedImage[ ] _idle_right
- BufferedImage[ ] _idle_left
- BufferedImage[ ] _walk_up
- BufferedImage[ ] _walk_down
- BufferedImage[ ] _walk_right
- BufferedImage[ ] _walk_left

Additional Inherited Members

8.4.1 Detailed Description

Represents an abstract character entity with position, hitbox, stats, and animations.

Definition at line 25 of file Character.java.

8.4.2 Constructor & Destructor Documentation

8.4.2.1 Character()

```
entity.Character.Character (
            String entityName,
            int x,
            int y,
            int dirX,
            int dirY,
            int speed,
            String facing,
            int _spriteCntMax,
            int spriteSpeed )
```

Constructor for the Character class.

Parameters

x	The initial X-coordinate of the character in the world.
y	The initial Y-coordinate of the character in the world.
dirX	The X-direction of the character.
dirY	The Y-direction of the character.
speed	The speed of the character's movement.
facing	The direction the character is facing (up, down, left, right).
_spriteCntMax	The maximum number of sprites for animation.
spriteSpeed	The speed of sprite animation.

Definition at line 60 of file Character.java.

```
00060
    {
00061         super(entityName, x, y, _spriteCntMax, spriteSpeed,true);
00062
00063
00064
00065         // Hitbox settings (size of the entity)
00066         this.hitbox.width = Const.WRLD_entityScreenSize / 2;
00067         this.hitbox.height = Const.WRLD_entityScreenSize / 2;
00068         this.dirX = dirX;
00069         this.dirY = dirY;
00070         this.speed = speed;
00071         this.facing = facing;
00072
00073         _idle_up = new BufferedImage[_spriteCntMax];
00074         _idle_down = new BufferedImage[_spriteCntMax];
00075         _idle_right = new BufferedImage[_spriteCntMax];
00076         _idle_left = new BufferedImage[_spriteCntMax];
00077         _walk_up = new BufferedImage[_spriteCntMax];
00078         _walk_down = new BufferedImage[_spriteCntMax];
00079         _walk_right = new BufferedImage[_spriteCntMax];
00080         _walk_left = new BufferedImage[_spriteCntMax];
00081         loadTextures(entityName);
00082     }
```

Here is the call graph for this function:

8.4.3 Member Function Documentation

8.4.3.1 accelerate()

```
void entity.Character.accelerate (
            int maxSpeed,
            int factor,
            double dt )  [protected]
```

Accelerates the character's speed up to the specified maximum speed.

Parameters

maxSpeed	The maximum speed to accelerate to.
dt	The time elapsed since the last update.

Definition at line 168 of file Character.java.

```
00168                                                                    {
00169         if (speed < maxSpeed) {
00170             speed += factor*dt;
00171         }
00172         if (speed > maxSpeed) {
00173             speed = maxSpeed;
00174         }
00175     }
```

Here is the caller graph for this function:

8.4.3.2 checkTileCollision()

```
void entity.Character.checkTileCollision (
            TileManager tileManager )  [private]
```

Checks for collision with nearby tiles using the character's hitbox.

Parameters

tileManager	The TileManager containing information about tiles in the world.

Definition at line 111 of file Character.java.

```
00111                                                       {
00112         // Checking tiles with hitbox
00113
00114
00115         if((tileManager.getTile(hitbox.x, hitbox.y - 5).getCollision() //Checks collision with tile on
      top of the character
00116         || tileManager.getTile(hitbox.x + hitbox.width , hitbox.y - 5).getCollision() )&&  dirY == -1)
      {
00117             worldY = tileManager.getTile(hitbox.x,hitbox.y).getPos()[1] - hitbox.height;    //Prevent
      moving if collidable terrain
00118         }
00119         if((tileManager.getTile(hitbox.x, hitbox.y + hitbox.height + 5).getCollision() //Checks
      collision with tile beneath of the character
00120         || tileManager.getTile(hitbox.x + hitbox.width, hitbox.y + hitbox.height + 5).getCollision())
      && dirY == 1){
00121             worldY = tileManager.getTile(hitbox.x,hitbox.y).getPos()[1] -1;        //Prevent moving if
      collidable terrain
00122         }
00123         if((tileManager.getTile(hitbox.x - 5, hitbox.y).getCollision() //Checks collision with tile on
      the left of the character
00124         || tileManager.getTile(hitbox.x - 5, hitbox.y + hitbox.height).getCollision()) && dirX == -1)
      {
00125             worldX = tileManager.getTile(hitbox.x,hitbox.y).getPos()[0] - hitbox.width/2 ;
      //Prevent moving if collidable terrain
00126         }
00127         if((tileManager.getTile(hitbox.x + hitbox.width + 5, hitbox.y).getCollision() //Checks
      collision with tile on the right of the character
00128         || tileManager.getTile(hitbox.x + hitbox.width + 5, hitbox.y + hitbox.height).getCollision())
      && dirX == 1){
00129             worldX = tileManager.getTile(hitbox.x,hitbox.y).getPos()[0] + hitbox.width/2 - 1;
      //Prevent moving if collidable terrain
00130         }
00131     }
```

Here is the call graph for this function: Here is the caller graph for this function:

8.4.3.3 decelerate()

```
void entity.Character.decelerate (
             int factor,
             double dt )  [protected]
```

Decelerates the character's speed.

Parameters

dt	The time elapsed since the last update.

Definition at line 181 of file Character.java.

```
00181                                                 {
00182         speed -= factor*dt;
00183     }
```

Here is the caller graph for this function:

8.4.3.4 drawInFight()

```
void entity.Character.drawInFight (
             Graphics2D g2,
```

```
            int screenX,
            int screenY )
```

Draws the character in the fight scene.

Parameters

g2	The Graphics2D object for drawing.
screenX	The X-coordinate on the screen.
screenY	The Y-coordinate on the screen.

Definition at line 276 of file Character.java.

```
00276                                                                   {
00277         // Other function to draw in fight scene
00278     }
```

Here is the caller graph for this function:

8.4.3.5 drawInWorld()

```
void entity.Character.drawInWorld (
            Graphics2D g2,
            int screenX,
            int screenY )
```

Draws the character in the world scene.

Parameters

g2	The Graphics2D object for drawing.
screenX	The X-coordinate on the screen.
screenY	The Y-coordinate on the screen.

Definition at line 215 of file Character.java.

```
00215                                                                   {
00216         BufferedImage image = null;
00217         if (speed == 0) { // IDLE ANIMATIONS
00218             for (int i = 0; i < _spriteCntMax; i++) {
00219                 switch (facing) {
00220                     case "up":
00221                         if (_spriteCnt == i) image = _idle_up[i];
00222                         break;
00223                     case "down":
00224                         if (_spriteCnt == i) image = _idle_down[i];
00225                         break;
00226                     case "left":
00227                         if (_spriteCnt == i) image = _idle_left[i];
00228                         break;
00229                     case "right":
00230                         if (_spriteCnt == i) image = _idle_right[i];
00231                         break;
00232                 }
00233             }
00234         }
00235         if (speed > 0) { // WALKING ANIMATIONS
00236             for (int i = 0; i < _spriteCntMax; i++) {
00237                 switch (facing) {
00238                     case "up":
00239                         if (_spriteCnt == i) image = _walk_up[i];
00240                         break;
00241                     case "down":
00242                         if (_spriteCnt == i) image = _walk_down[i];
00243                         break;
00244                     case "left":
```

```
00245                         if (_spriteCnt == i) image = _walk_left[i];
00246                         break;
00247                     case "right":
00248                         if (_spriteCnt == i) image = _walk_right[i];
00249                         break;
00250                 }
00251             }
00252         }
00253 
00254 
00255         int playerScreenX = (Const.WDW_width - Const.WRLD_entityScreenSize) / 2;
00256         int playerScreenY = (Const.WDW_height - Const.WRLD_entityScreenSize) / 2;
00257 
00258         //Checking if we need to draw enemy or not
00259         if (worldX + Const.WRLD_tileScreenSize > worldX - playerScreenX
00260         && worldX - Const.WRLD_tileScreenSize < worldX + playerScreenX
00261         && worldY + Const.WRLD_tileScreenSize > worldY - playerScreenY
00262         && worldY - Const.WRLD_tileScreenSize < worldY + playerScreenY) {
00263 
00264             g2.drawImage(image, screenX, screenY, Const.WRLD_entityScreenSize,
      Const.WRLD_entityScreenSize, null);
00265             g2.drawRect(screenX + hitbox.width / 2, screenY + hitbox.height, hitbox.width,
      hitbox.height); // Center the hitbox to the entity
00266         }
00267     }
```

Here is the caller graph for this function:

8.4.3.6 loadTextures()

```
void entity.Character.loadTextures (
            String name )  [protected]
```

Loads character textures based on the specified name.

Parameters

name	The name used to determine the textures to load.

Definition at line 189 of file Character.java.

```
00189                                           {
00190         try {
00191             for (int i = 0; i < _spriteCntMax; i++) {
00192                 _idle_up[i] = ImageIO.read(new FileInputStream("res/entity/character/idle/" + name +
      "/up" + (i + 1) + ".png"));
00193                 _idle_down[i] = ImageIO.read(new FileInputStream("res/entity/character/idle/" + name +
      "/down" + (i + 1) + ".png"));
00194                 _idle_left[i] = ImageIO.read(new FileInputStream("res/entity/character/idle/" + name +
      "/left" + (i + 1) + ".png"));
00195                 _idle_right[i] = ImageIO.read(new FileInputStream("res/entity/character/idle/" + name
      + "/right" + (i + 1) + ".png"));
00196 
00197                 _walk_up[i] = ImageIO.read(new FileInputStream("res/entity/character/walk/" + name +
      "/up" + (i + 1) + ".png"));
00198                 _walk_down[i] = ImageIO.read(new FileInputStream("res/entity/character/walk/" + name +
      "/down" + (i + 1) + ".png"));
00199                 _walk_left[i] = ImageIO.read(new FileInputStream("res/entity/character/walk/" + name +
      "/left" + (i + 1) + ".png"));
00200                 _walk_right[i] = ImageIO.read(new FileInputStream("res/entity/character/walk/" + name
      + "/right" + (i + 1) + ".png"));
00201             }
00202 
00203         } catch (IOException e) {
00204             e.printStackTrace();
00205         }
00206     }
```

8.4.3.7 move()

```
void entity.Character.move (
            World world,
```

```
            int speed,
            double dt )  [protected]
```

Moves the character in the world based on its direction, speed, and the elapsed time.

Parameters

world	The current game world.
speed	The speed of the character's movement.
dt	The time elapsed since the last update.

Definition at line 141 of file Character.java.

```
00141                                                                {
00142         if ((dirX == 0 && dirY != 0) || (dirY == 0 && dirX != 0)) {
00143             if (dirX == 1) {
00144                 worldX += speed * dt;
00145             }
00146             if (dirX == -1) {
00147                 worldX -= speed * dt;
00148             }
00149             if (dirY == 1) {
00150                 worldY += speed * dt;
00151             }
00152             if (dirY == -1) {
00153                 worldY -= speed * dt;
00154             }
00155         }
00156         if (dirX != 0 && dirY != 0) {
00157             double normSum = Math.sqrt(dirX * dirX + dirY * dirY);      //Normalizing vector
00158             worldX += (dirX / normSum) * speed * dt;
00159             worldY += (dirY / normSum) * speed * dt;
00160         }
00161     }
```

Here is the caller graph for this function:

8.4.3.8 playerInterraction()

```
void entity.Character.playerInterraction (
            Player player )  [protected]
```

Reimplemented from entity.Entity.

Reimplemented in entity.Enemy.

Definition at line 281 of file Character.java.

```
00281                                                {
00282     }
```

8.4.3.9 update()

```
void entity.Character.update (
            Scene scene,
            double dt )
```

Updates the character entity based on the current scene and time elapsed.

Parameters

scene	The current game scene.
dt	The time elapsed since the last update.

Reimplemented from entity.Entity.

Reimplemented in entity.Enemy, and entity.Player.

Definition at line 90 of file Character.java.

```
00090                                           {
00091         if (scene.state == State.WORLD) {
00092             World currWorld = World.getWorld();
00093 
00094             hitbox.x = worldX + hitbox.width / 2;
00095             hitbox.y = worldY + hitbox.height;
00096 
00097             // CHECK THE COLLISION
00098             move(World.getWorld(), speed, dt);
00099             checkTileCollision(currWorld.tileManager);
00100 
00101 
00102 
00103             updateFrames();
00104         }
00105     }
```

Here is the call graph for this function:

8.4.4 Member Data Documentation

8.4.4.1 _idle_down

```
BufferedImage [] entity.Character._idle_down  [private]
```

Definition at line 41 of file Character.java.

8.4.4.2 _idle_left

```
BufferedImage [] entity.Character._idle_left  [private]
```

Definition at line 43 of file Character.java.

8.4.4.3 _idle_right

```
BufferedImage [] entity.Character._idle_right  [private]
```

Definition at line 42 of file Character.java.

8.4.4.4 _idle_up

```
BufferedImage [] entity.Character._idle_up  [private]
```

Definition at line 40 of file Character.java.

8.4.4.5 _walk_down

```
BufferedImage [] entity.Character._walk_down  [private]
```

Definition at line 45 of file Character.java.

8.4.4.6 _walk_left

```
BufferedImage [] entity.Character._walk_left  [private]
```

Definition at line 47 of file Character.java.

8.4.4.7 _walk_right

```
BufferedImage [] entity.Character._walk_right  [private]
```

Definition at line 46 of file Character.java.

8.4.4.8 _walk_up

```
BufferedImage [] entity.Character._walk_up  [private]
```

Definition at line 44 of file Character.java.

8.4.4.9 agility

```
int entity.Character.agility  [package]
```

Definition at line 28 of file Character.java.

8.4.4.10 defense

```
int entity.Character.defense  [package]
```

Definition at line 30 of file Character.java.

8.4.4.11 dirX

```
int entity.Character.dirX
```

Definition at line 33 of file Character.java.

8.4.4.12 dirY

```
int entity.Character.dirY
```

Definition at line 33 of file Character.java.

8.4.4.13 facing

```
String entity.Character.facing
```

Definition at line 37 of file Character.java.

8.4.4.14 hasKey

```
int entity.Character.hasKey = 0  [package]
```

Definition at line 34 of file Character.java.

8.4.4.15 health

```
int entity.Character.health  [package]
```

Definition at line 26 of file Character.java.

8.4.4.16 initiative

```
int entity.Character.initiative  [package]
```

Definition at line 31 of file Character.java.

8.4.4.17 mana

```
int entity.Character.mana  [package]
```

Definition at line 27 of file Character.java.

8.4.4.18 speed

```
int entity.Character.speed
```

Definition at line 32 of file Character.java.

8.4.4.19 strength

```
int entity.Character.strength  [package]
```

Definition at line 29 of file Character.java.

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

- src/entity/Character.java

8.5 UI.ChoiceButton Class Reference

Collaboration diagram for UI.ChoiceButton:

Public Member Functions

- ChoiceButton (int w, int h, String title, String fontName, Color fontColor)
- void draw (Graphics2D g2, int x, int y)

Public Attributes

- int width
- int height

Private Member Functions

- void loadTexture ()

Private Attributes

- Textbox _textBox
- BufferedImage _bgTexture

8.5.1 Detailed Description

Definition at line 13 of file ChoiceButton.java.

8.5.2 Constructor & Destructor Documentation

8.5.2.1 ChoiceButton()

```
UI.ChoiceButton.ChoiceButton (
            int w,
            int h,
            String title,
            String fontName,
            Color fontColor )
```

Definition at line 18 of file ChoiceButton.java.

```
00018                                                                                {
00019         this.width = w; this.height = h;
00020
00021         _textBox = new Textbox(title,fontName,w,h,fontColor);
00022
00023         loadTexture();
00024     }
```

Here is the call graph for this function:

8.5.3 Member Function Documentation

8.5.3.1 draw()

```
void UI.ChoiceButton.draw (
            Graphics2D g2,
            int x,
            int y )
```

Definition at line 35 of file ChoiceButton.java.

```
00035                                                {
00036         //g2.drawImage(_bgTexture,x,y,80*3,20*3,null);
00037         g2.setColor(new Color(0xA38168));
00038         g2.fillRect(x,y,width,height);
00039         _textBox.draw(g2,x,y);
00040     }
```

Here is the call graph for this function: Here is the caller graph for this function:

8.5.3.2 loadTexture()

```
void UI.ChoiceButton.loadTexture ( )  [private]
```

Definition at line 26 of file ChoiceButton.java.

```
00026                                {
00027         try{
00028             _bgTexture = ImageIO.read(new FileInputStream("res/hud/bg.png"));
00029         }
00030         catch(IOException e){
00031             e.printStackTrace();
00032         }
00033     }
```

Here is the caller graph for this function:

8.5.4 Member Data Documentation

8.5.4.1 _bgTexture

```
BufferedImage UI.ChoiceButton._bgTexture  [private]
```

Definition at line 16 of file ChoiceButton.java.

8.5.4.2 _textBox

```
Textbox UI.ChoiceButton._textBox  [private]
```

Definition at line 15 of file ChoiceButton.java.

8.5.4.3 height

```
int UI.ChoiceButton.height
```

Definition at line 14 of file ChoiceButton.java.

8.5.4.4 width

```
int UI.ChoiceButton.width
```

Definition at line 14 of file ChoiceButton.java.

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

- src/UI/ChoiceButton.java

8.6 game.Const Class Reference

Collaboration diagram for game.Const:

Static Public Attributes

- static final int WDW_width = 800
- static final int WDW_height = 600
- static final int WRLD_tileSize = 16
- static final int WRLD_scale = 3
- static final int WRLD_tileScreenSize = WRLD_tileSize∗WRLD_scale
- static final int WRLD_entityScreenSize = WRLD_tileSize∗WRLD_scale
- static final int WRLD_maxRow = 27
- static final int WRLD_maxCol = 27
- static final int FGHT_entityScreenSize = 200
- static final int nbFloorTextures = 6
- static final int nbTopTextures = 29

8.6.1 Detailed Description

Definition at line 3 of file Const.java.

8.6.2 Member Data Documentation

8.6.2.1 FGHT_entityScreenSize

```
final int game.Const.FGHT_entityScreenSize = 200  [static]
```

Definition at line 18 of file Const.java.

8.6.2.2 nbFloorTextures

```
final int game.Const.nbFloorTextures = 6  [static]
```

Definition at line 21 of file Const.java.

8.6.2.3 nbTopTextures

```
final int game.Const.nbTopTextures = 29  [static]
```

Definition at line 22 of file Const.java.

8.6.2.4 WDW_height

```
final int game.Const.WDW_height = 600  [static]
```

Definition at line 7 of file Const.java.

8.6.2.5 WDW_width

```
final int game.Const.WDW_width = 800  [static]
```

Definition at line 6 of file Const.java.

8.6.2.6 WRLD_entityScreenSize

```
final int game.Const.WRLD_entityScreenSize = WRLD_tileSize*WRLD_scale  [static]
```

Definition at line 13 of file Const.java.

8.6.2.7 WRLD_maxCol

```
final int game.Const.WRLD_maxCol = 27  [static]
```

Definition at line 15 of file Const.java.

8.6.2.8 WRLD_maxRow

```
final int game.Const.WRLD_maxRow = 27  [static]
```

Definition at line 15 of file Const.java.

8.6.2.9 WRLD_scale

```
final int game.Const.WRLD_scale = 3  [static]
```

Definition at line 11 of file Const.java.

8.6.2.10 WRLD_tileScreenSize

```
final int game.Const.WRLD_tileScreenSize = WRLD_tileSize*WRLD_scale  [static]
```

Definition at line 12 of file Const.java.

8.6.2.11 WRLD_tileSize

```
final int game.Const.WRLD_tileSize = 16  [static]
```

Definition at line 10 of file Const.java.

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

- src/game/Const.java

8.7 entity.Enemy Class Reference

Represents an enemy entity in the game.

Inheritance diagram for entity.Enemy:

Collaboration diagram for entity.Enemy:

Public Member Functions

- Enemy (String enemyName, int worldX, int worldY, int dirX, int dirY, int speed, String facing, int spriteCntMax, int spriteSpeed)

  Constructor for the Enemy class.
- boolean touchingPlayer (Player player)

  Checks if the current enemy is in contact with the player.
- void update (Scene scene, double dt)

  Updates the enemy entity based on the current scene and time elapsed.
- void playerInterraction (Player player)

Public Attributes

- String name

Private Attributes

- int _xpRate

Additional Inherited Members

8.7.1 Detailed Description

Represents an enemy entity in the game.

Definition at line 15 of file Enemy.java.

8.7.2 Constructor & Destructor Documentation

8.7.2.1 Enemy()

```
entity.Enemy.Enemy (
            String enemyName,
            int worldX,
            int worldY,
            int dirX,
            int dirY,
            int speed,
            String facing,
            int spriteCntMax,
            int spriteSpeed )
```

Constructor for the Enemy class.

Parameters

worldX	The X-coordinate of the enemy in the world.
worldY	The Y-coordinate of the enemy in the world.
dirX	The X-direction of the enemy.
dirY	The Y-direction of the enemy.
speed	The speed of the enemy's movement.
facing	The direction the enemy is facing (up, down, left, right).
spriteCntMax	The maximum number of sprites for animation.
spriteSpeed	The speed of sprite animation.

Definition at line 30 of file Enemy.java.

```
00030
      {
00031         super(enemyName, worldX, worldY, dirX, dirY, speed, facing, spriteCntMax, spriteSpeed);  //
      Calls the parent class for entity setup, specifying scene.keyH for player
00032     }
```

8.7.3 Member Function Documentation

8.7.3.1 playerInterraction()

```
void entity.Enemy.playerInterraction (
            Player player )
```

Reimplemented from entity.Character.

Definition at line 62 of file Enemy.java.

```
00062                                           {
00063
00064     }
```

8.7.3.2 touchingPlayer()

```
boolean entity.Enemy.touchingPlayer (
            Player player )
```

Checks if the current enemy is in contact with the player.

Parameters

player	The player entity

Definition at line 38 of file Enemy.java.

```
00038                                           {
00039
00040         if((hitbox.x >= player.hitbox.x + player.hitbox.width)      // trop à droite
00041         || (hitbox.x + hitbox.width <= player.hitbox.x) // trop à gauche
00042         || (hitbox.y >= player.hitbox.y + player.hitbox.height) // trop en bas
00043         || (hitbox.y + hitbox.height <= player.hitbox.y)){// trop en haut
00044             return false;
00045         }
00046
00047         return true;
00048     }
```

8.7.3.3 update()

```
void entity.Enemy.update (
            Scene scene,
            double dt )
```

Updates the enemy entity based on the current scene and time elapsed.

Parameters

scene	The current game scene.
dt	The time elapsed since the last update.

Reimplemented from entity.Character.

Definition at line 57 of file Enemy.java.

```
00057                                           {
00058         super.update(scene, dt); // Calls the parent class update method
00059         //TODO : find a method to make the enemy move in predictive patterns
00060     }
```

8.7.4 Member Data Documentation

8.7.4.1 _xpRate

```
int entity.Enemy._xpRate  [private]
```

Definition at line 17 of file Enemy.java.

8.7.4.2 name

```
String entity.Enemy.name
```

Definition at line 18 of file Enemy.java.

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

- src/entity/Enemy.java

8.8 entity.Entity Class Reference

Represents an abstract entity with position, hitbox, and animations.

Inheritance diagram for entity.Entity:

Collaboration diagram for entity.Entity:

Public Member Functions

- Entity ()

  Default constructor for the Entity class.
- Entity (String entityName, int x, int y, int _spriteCntMax, int spriteSpeed, boolean collision)

  Constructor for the Entity class with specified initial position and animation parameters.
- void update (Scene scene, double dt)

  Updates the entity's position based on the current scene and time elapsed.

Public Attributes

- int worldX
- int worldY
- Rectangle hitbox = new Rectangle()
- String name

Protected Member Functions

- void updateFrames ()

  Updates the frames of the entity's animation.
- abstract void playerInterraction (Player player)

Protected Attributes

- boolean collision
- int _spriteCnt = 0
- int _spriteUpdater = 0
- int _spriteSpeed
- int _spriteCntMax

Private Member Functions

- void loadTextures ()

  Loads textures for the entity based on its name.

8.8.1 Detailed Description

Represents an abstract entity with position, hitbox, and animations.

Definition at line 15 of file Entity.java.

8.8.2 Constructor & Destructor Documentation

8.8.2.1 Entity() [1/2]

```
entity.Entity.Entity ( )
```

Default constructor for the Entity class.

Definition at line 33 of file Entity.java.

```
00033                     {
00034         // Default constructor
00035     }
```

8.8.2.2 Entity() [2/2]

```
entity.Entity.Entity (
            String entityName,
            int x,
            int y,
            int _spriteCntMax,
            int spriteSpeed,
            boolean collision )
```

Constructor for the Entity class with specified initial position and animation parameters.

Parameters

x	The initial X-coordinate of the entity in the world.
y	The initial Y-coordinate of the entity in the world.
_spriteCntMax	The maximum number of sprites for animation.
spriteSpeed	The speed of sprite animation.

Definition at line 44 of file Entity.java.

```
00044
      {
00045         this.collision = collision;
00046         this.worldX = x;
00047         this.worldY = y;
00048         this.hitbox.x = worldX;
00049         this.hitbox.y = worldY;
00050         this._spriteCntMax = _spriteCntMax;
00051         this._spriteSpeed = spriteSpeed;
00052         this.name = entityName;
00053     }
```

8.8.3 Member Function Documentation

8.8.3.1 loadTextures()

```
void entity.Entity.loadTextures ( )  [private]
```

Loads textures for the entity based on its name.

Parameters

name	The name used to determine the textures to load.

Definition at line 70 of file Entity.java.

```
00070                                {
00071         // TODO: Different texture loading from characters
00072     }
```

Here is the caller graph for this function:

8.8.3.2 playerInterraction()

```
abstract void entity.Entity.playerInterraction (
             Player player )  [abstract], [protected]
```

Reimplemented in entity.props.OBJ_Chest, entity.props.OBJ_Door, entity.props.OBJ_Key, entity.Character, and entity.Enemy.

8.8.3.3 update()

```
void entity.Entity.update (
            Scene scene,
            double dt )
```

Updates the entity's position based on the current scene and time elapsed.

Parameters

scene	The current game scene.
dt	The time elapsed since the last update.

Reimplemented in entity.Character, entity.Enemy, and entity.Player.

Definition at line 60 of file Entity.java.

```
00060                                           {
00061         // Updating entity position accurately (at any point in time either pressing keys or not)
00062     }
```

8.8.3.4 updateFrames()

```
void entity.Entity.updateFrames ( )  [protected]
```

Updates the frames of the entity's animation.

Definition at line 77 of file Entity.java.

```
00077                                   {
00078         _spriteUpdater++;
00079         if (_spriteUpdater > _spriteSpeed) {
00080             _spriteCnt++;
00081             if (_spriteCnt == _spriteCntMax) {
00082                 _spriteCnt = 0;
00083             }
00084             _spriteUpdater = 0;
00085         }
00086     }
```

Here is the caller graph for this function:

8.8.4 Member Data Documentation

8.8.4.1 _spriteCnt

```
int entity.Entity._spriteCnt = 0  [protected]
```

Definition at line 23 of file Entity.java.

8.8.4.2 _spriteCntMax

```
int entity.Entity._spriteCntMax  [protected]
```

Definition at line 26 of file Entity.java.

8.8.4.3 _spriteSpeed

```
int entity.Entity._spriteSpeed  [protected]
```

Definition at line 25 of file Entity.java.

8.8.4.4 _spriteUpdater

```
int entity.Entity._spriteUpdater = 0  [protected]
```

Definition at line 24 of file Entity.java.

8.8.4.5 collision

```
boolean entity.Entity.collision  [protected]
```

Definition at line 18 of file Entity.java.

8.8.4.6 hitbox

```
Rectangle entity.Entity.hitbox = new Rectangle()
```

Definition at line 20 of file Entity.java.

8.8.4.7 name

```
String entity.Entity.name
```

Definition at line 28 of file Entity.java.

8.8.4.8 worldX

```
int entity.Entity.worldX
```

Definition at line 17 of file Entity.java.

8.8.4.9 worldY

```
int entity.Entity.worldY
```

Definition at line 17 of file Entity.java.

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

- src/entity/Entity.java

8.9 entity.EntitySetter Class Reference

Responsible for initializing and setting up objects (entities) in the game world.

Collaboration diagram for entity.EntitySetter:

Public Member Functions

- EntitySetter (World world)

  Constructor for the EntitySetter class.
- void setObject ()

  Method to set up objects in the game world, such as keys, doors, and chests.
- void setEnemies ()

Package Attributes

- World world

8.9.1 Detailed Description

Responsible for initializing and setting up objects (entities) in the game world.

Definition at line 17 of file EntitySetter.java.

8.9.2 Constructor & Destructor Documentation

8.9.2.1 EntitySetter()

```
entity.EntitySetter.EntitySetter (
            World world )
```

Constructor for the EntitySetter class.

Parameters

world	The World object representing the game world.

Definition at line 24 of file EntitySetter.java.

```
00024                                       {
00025         this.world = world;
00026     }
```

8.9.3 Member Function Documentation

8.9.3.1 setEnemies()

```
void entity.EntitySetter.setEnemies ( )
```

Definition at line 45 of file EntitySetter.java.

```
00045                             {
00046         Enemy enemy1 = new Enemy("orc", 8*Const.WRLD_entityScreenSize, 10*Const.WRLD_entityScreenSize,
      0, 0, 0, "down", 4, 20);
00047         world.addEnemy(new Point((int) enemy1.worldX, (int) enemy1.worldY), enemy1);
00048
00049         Enemy enemy2 = new Enemy("orc", 16*Const.WRLD_entityScreenSize,
      10*Const.WRLD_entityScreenSize, 0, 0, 0, "up", 4, 20);
00050         world.addEnemy(new Point((int) enemy2.worldX, (int) enemy2.worldY), enemy2);
00051
00052     }
```

Here is the call graph for this function: Here is the caller graph for this function:

8.9.3.2 setObject()

```
void entity.EntitySetter.setObject ( )
```

Method to set up objects in the game world, such as keys, doors, and chests.

Definition at line 31 of file EntitySetter.java.

```
00031                             {
00032         // Create and set up a Key object at a specific location in the world
00033         Props key = new OBJ_Key(13 * Const.WRLD_entityScreenSize,13 * Const.WRLD_entityScreenSize);
00034         world.addObject(new Point((int) key.worldX, (int) key.worldY), key);
00035
00036         // Create and set up a Door object at a specific location in the world
00037         Props door = new OBJ_Door(14 * Const.WRLD_entityScreenSize,13 * Const.WRLD_entityScreenSize);
00038         world.addObject(new Point((int) door.worldX, (int) door.worldY), door);
00039
00040         // Create and set up a Chest object at a specific location in the world
00041         Props chest = new OBJ_Chest(15 * Const.WRLD_entityScreenSize,13 *
      Const.WRLD_entityScreenSize);
00042         world.addObject(new Point((int) chest.worldX, (int) chest.worldY), chest);
00043     }
```

Here is the call graph for this function: Here is the caller graph for this function:

8.9.4 Member Data Documentation

8.9.4.1 world

```
World entity.EntitySetter.world  [package]
```

Definition at line 18 of file EntitySetter.java.

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

- src/entity/EntitySetter.java

8.10 game.FightScene Class Reference

Represents the scene during a fight between a player and an enemy.

Collaboration diagram for game.FightScene:

Classes

- enum FightState

Public Member Functions

- FightScene (Player player, Enemy enemy)

  Constructor for the FightScene class.
- void update (Scene scene)

  Updates the fight scene.
- void killEnemy (HashMap< Point, Enemy > enemies, Enemy enemy)
- void draw (Graphics2D g2)

  Draws the fight scene.

Public Attributes

- Player player
- Enemy enemy
- FightState state

Private Attributes

- HUD menu

8.10.1 Detailed Description

Represents the scene during a fight between a player and an enemy.

Definition at line 24 of file FightScene.java.

8.10.2 Constructor & Destructor Documentation

8.10.2.1 FightScene()

```
game.FightScene.FightScene (
            Player player,
            Enemy enemy )
```

Constructor for the FightScene class.

Parameters

player	The player entity in the fight.
enemy	The enemy entity in the fight.

Definition at line 37 of file FightScene.java.

```
00037                                                   {
00038         System.out.println("Entering combat");
00039         this.player = player;
00040         this.enemy = enemy;
00041         state = FightState.FIGHTING;
00042         menu = new HUD(MenuType.FIGHT);
00043     }
```

8.10.3 Member Function Documentation

8.10.3.1 draw()

```
void game.FightScene.draw (
            Graphics2D g2 )
```

Draws the fight scene.

Parameters

g2	The Graphics2D object for drawing.
screenWidth	The width of the screen.
screenHeight	The height of the screen.

Definition at line 73 of file FightScene.java.

```
00073                                       {
00074         g2.setColor( new Color(0xFF2265));
00075         g2.fillRect(100,200,400,150);
00076         player.drawInFight(g2, Const.WDW_width / 2 - (Const.FGHT_entityScreenSize / 2),
      Const.WDW_height / 2 - (Const.FGHT_entityScreenSize / 2));
00077         menu.draw(g2, Const.WDW_width, Const.WDW_height);
00078     }
```

Here is the call graph for this function: Here is the caller graph for this function:

8.10.3.2 killEnemy()

```
void game.FightScene.killEnemy (
            HashMap< Point, Enemy > enemies,
            Enemy enemy )
```

Definition at line 62 of file FightScene.java.

```
00062                                                                   {
00063         enemies.remove(new Point(enemy.worldX,enemy.worldY), enemy);
00064     }
```

Here is the caller graph for this function:

8.10.3.3 update()

```
void game.FightScene.update (
            Scene scene )
```

Updates the fight scene.

Definition at line 48 of file FightScene.java.

```
00048                                   {
00049
00050         // Additional logic for the fight scene update
00051         System.out.println("Le joueur est en combat avec "+ enemy.name);
00052         if(scene.keyH.interactPressed){
00053             state = FightState.WON;
00054             scene.state = State.WORLD;
00055             player.speed = 0;
00056             killEnemy(World.enemies, enemy);
00057
00058             scene.keyH.interactPressed = false;
00059         }
00060     }
```

Here is the call graph for this function: Here is the caller graph for this function:

8.10.4 Member Data Documentation

8.10.4.1 enemy

```
Enemy game.FightScene.enemy
```

Definition at line 28 of file FightScene.java.

8.10.4.2 menu

```
HUD game.FightScene.menu  [private]
```

Definition at line 30 of file FightScene.java.

8.10.4.3 player

```
Player game.FightScene.player
```

Definition at line 27 of file FightScene.java.

8.10.4.4 state

```
FightState game.FightScene.state
```

Definition at line 29 of file FightScene.java.

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

- src/game/FightScene.java

8.11 game.FightScene.FightState Enum Reference

Collaboration diagram for game.FightScene.FightState:

Public Attributes

- FIGHTING
- WON
- LOST

8.11.1 Detailed Description

Definition at line 25 of file FightScene.java.

8.11.2 Member Data Documentation

8.11.2.1 FIGHTING

```
game.FightScene.FightState.FIGHTING
```

Definition at line 25 of file FightScene.java.

8.11.2.2 LOST

```
game.FightScene.FightState.LOST
```

Definition at line 25 of file FightScene.java.

8.11.2.3 WON

`game.FightScene.FightState.WON`

Definition at line 25 of file FightScene.java.

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

- src/game/FightScene.java

8.12 item.armor.Foot Class Reference

Inheritance diagram for item.armor.Foot:

Collaboration diagram for item.armor.Foot:

Public Member Functions

- Foot ()

8.12.1 Detailed Description

Definition at line 3 of file Foot.java.

8.12.2 Constructor & Destructor Documentation

8.12.2.1 Foot()

`item.armor.Foot.Foot ( )`

Definition at line 5 of file Foot.java.

```
00005                 {
00006         System.out.println("Armure foot");
00007     }
```

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

- src/item/armor/Foot.java

8.13 item.Generator Class Reference

Collaboration diagram for item.Generator:

Static Public Member Functions

- static Item generateItem ()
- static Weapon generateWeapon ()
- static Potion generatePotion ()
- static Armor generateArmor ()

8.13.1 Detailed Description

Definition at line 7 of file Generator.java.

8.13.2 Member Function Documentation

8.13.2.1 generateArmor()

```
static Armor item.Generator.generateArmor ( )  [static]
```

Definition at line 36 of file Generator.java.

```
00036                                       {
00037         int nbRandom= (int) (Math.random() * 4);
00038         return switch (nbRandom) {
00039             case 1 -> new Body();
00040             case 2 -> new Head();
00041             case 3 -> new Legs();
00042             default -> new Foot();
00043         };
00044     }
```

Here is the caller graph for this function:

8.13.2.2 generateItem()

```
static Item item.Generator.generateItem ( )  [static]
```

Definition at line 9 of file Generator.java.

```
00009                                       {
00010         int nbRandom= (int) (Math.random() * 3);
00011         return switch (nbRandom) {
00012             case 1 -> Generator.generateWeapon();
00013             case 2 -> Generator.generateArmor();
00014             default -> Generator.generatePotion();
00015         };
00016     }
```

Here is the call graph for this function: Here is the caller graph for this function:

8.13.2.3 generatePotion()

```
static Potion item.Generator.generatePotion ( )  [static]
```

Definition at line 27 of file Generator.java.

```
00027                                       {
00028         int nbRandom= (int) (Math.random() * 3);
00029         return switch (nbRandom) {
00030             case 1 -> new HealthPotion();
00031             case 2 -> new ManaPotion();
00032             default -> new SpeedPotion();
00033         };
00034     }
```

Here is the caller graph for this function:

8.13.2.4 generateWeapon()

```
static Weapon item.Generator.generateWeapon ( )  [static]
```

Definition at line 18 of file Generator.java.

```
00018                                       {
00019         int nbRandom= (int) (Math.random() * 3);
00020         return switch (nbRandom) {
00021             case 1 -> new Bow();
00022             case 2 -> new Staff();
00023             default -> new Sword();
00024         };
00025     }
```

Here is the caller graph for this function:

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

- src/item/Generator.java

8.14 item.armor.Head Class Reference

Inheritance diagram for item.armor.Head:

Collaboration diagram for item.armor.Head:

Public Member Functions

- Head ()

8.14.1 Detailed Description

Definition at line 3 of file Head.java.

8.14.2 Constructor & Destructor Documentation

8.14.2.1 Head()

```
item.armor.Head.Head ( )
```

Definition at line 4 of file Head.java.

```
00004                 {
00005         System.out.println("Armure tete");
00006     }
```

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

- src/item/armor/Head.java

8.15 item.potion.HealthPotion Class Reference

Inheritance diagram for item.potion.HealthPotion:

Collaboration diagram for item.potion.HealthPotion:

Public Member Functions

- HealthPotion ()

8.15.1 Detailed Description

Definition at line 3 of file HealthPotion.java.

8.15.2 Constructor & Destructor Documentation

8.15.2.1 HealthPotion()

```
item.potion.HealthPotion.HealthPotion ( )
```

Definition at line 5 of file HealthPotion.java.

```
00005                           {
00006         System.out.println("Potion de soin");
00007     }
```

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

- src/item/potion/HealthPotion.java

8.16 UI.HUD Class Reference

Collaboration diagram for UI.HUD:

Classes

- enum MenuType

Public Member Functions

- HUD (MenuType type)
- void draw (Graphics2D g2, int screenWidth, int screenHeight)
- void update (double dt)

Public Attributes

- final int HUDWidth = 600
- final int HUDHeight = 550

Private Attributes

- int _nbButtons
- ChoiceButton[ ] _buttons
- ChoiceButton _title
- MenuType _type
- int _titleWidth
- int _titleHeight

8.16.1 Detailed Description

Definition at line 6 of file HUD.java.

8.16.2 Constructor & Destructor Documentation

8.16.2.1 HUD()

```
UI.HUD.HUD (
            MenuType type )
```

Definition at line 20 of file HUD.java.

```
00020                                {
00021         _type = type;
00022         switch(_type){
00023             case WELCOME:
00024                 _nbButtons = 2;
00025                 break;
00026             case PAUSE:
00027                 _nbButtons = 3;
00028                 _titleWidth = 300;
00029                 _titleHeight = 80;
00030                 _title = new ChoiceButton(_titleWidth,_titleHeight,"PAUSE","rainyhearts",new
      Color(0x834317));
00031                 break;
00032             case FIGHT:
00033                 _titleWidth = 300;
00034                 _titleHeight = 60;
00035                 _nbButtons = 4;
00036                 _title = new ChoiceButton(_titleWidth,_titleHeight,"FIGHT","rainyhearts",new
      Color(0xF10516));
00037                 break;
00038             case SHOP:
00039                 _nbButtons = 4;
00040                 break;
00041         }
00042
00043         _buttons = new ChoiceButton[_nbButtons];
00044
00045         //To replace with the current names that we want depending on the MenuType
00046
00047         for(int i =0; i<_nbButtons ; i++){
00048             _buttons[i] = new ChoiceButton(80*3,20*3, "BUTTON " +i, "rainyhearts",Color.black);
00049         }
00050     }
```

8.16.3 Member Function Documentation

8.16.3.1 draw()

```
void UI.HUD.draw (
            Graphics2D g2,
            int screenWidth,
            int screenHeight )
```

Definition at line 53 of file HUD.java.

```
00053                                                                    {
00054         //g2.drawImage(_bgTexture, (800 - HUDWidth)/2,(600 - HUDHeight)/2,HUDWidth, HUDHeight, null);
00055         g2.setColor(new Color(0,0,0,20));
00056         g2.fillRect((screenWidth - HUDWidth)/2, (screenHeight - HUDHeight)/2,HUDWidth,HUDHeight);
      //"Drawing" HUD with soft background color
00057
00058
00059
00060
00061         int gap = (HUDHeight - _nbButtons*_buttons[0].height)/(_nbButtons+1);    //Small gap to space
      the buttons and keep them centered on screen
00062         _title.draw(g2,(screenWidth - _titleWidth)/2, screenHeight - HUDHeight);
00063         for(int i=0;i < _nbButtons; i++){
00064             _buttons[i].draw(g2,(screenWidth - _buttons[i].width)/2, (gap+gap*(i+1)/2 +
      _buttons[i].height*i + (screenHeight - HUDHeight)/2));
00065         }
00066     }
```

Here is the call graph for this function: Here is the caller graph for this function:

8.16.3.2 update()

```
void UI.HUD.update (
            double dt )
```

Definition at line 68 of file HUD.java.

```
00068                                  {
00069
00070     }
```

8.16.4 Member Data Documentation

8.16.4.1 _buttons

```
ChoiceButton [] UI.HUD._buttons  [private]
```

Definition at line 10 of file HUD.java.

8.16.4.2 _nbButtons

```
int UI.HUD._nbButtons  [private]
```

Definition at line 9 of file HUD.java.

8.16.4.3 _title

```
ChoiceButton UI.HUD._title  [private]
```

Definition at line 11 of file HUD.java.

8.16.4.4 _titleHeight

```
int UI.HUD._titleHeight  [private]
```

Definition at line 13 of file HUD.java.

8.16.4.5 _titleWidth

```
int UI.HUD._titleWidth  [private]
```

Definition at line 13 of file HUD.java.

8.16.4.6 _type

```
MenuType UI.HUD._type  [private]
```

Definition at line 12 of file HUD.java.

8.16.4.7 HUDHeight

```
final int UI.HUD.HUDHeight = 550
```

Definition at line 18 of file HUD.java.

8.16.4.8 HUDWidth

```
final int UI.HUD.HUDWidth = 600
```

Definition at line 17 of file HUD.java.

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

- src/UI/HUD.java

8.17 item.Item Class Reference

Inheritance diagram for item.Item:

Collaboration diagram for item.Item:

8.17.1 Detailed Description

Definition at line 3 of file Item.java.

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

- src/item/Item.java

8.18 main.KeyHandler Class Reference

Handles keyboard input using the singleton pattern.

Inheritance diagram for main.KeyHandler:

Collaboration diagram for main.KeyHandler:

Public Member Functions

- void keyTyped (KeyEvent e)
- void keyPressed (KeyEvent e)
- void keyReleased (KeyEvent e)

Static Public Member Functions

- static KeyHandler getInstance ()

  Gets the instance of the KeyHandler using the singleton pattern.

Public Attributes

- boolean upPressed
- boolean downPressed
- boolean leftPressed
- boolean rightPressed
- boolean interactPressed
- boolean escPressed

Static Public Attributes

- static KeyHandler instance

Private Member Functions

- KeyHandler ()

8.18.1 Detailed Description

Handles keyboard input using the singleton pattern.

Definition at line 16 of file KeyHandler.java.

8.18.2 Constructor & Destructor Documentation

8.18.2.1 KeyHandler()

```
main.KeyHandler.KeyHandler ( )  [private]
```

Definition at line 23 of file KeyHandler.java.

```
00023 {}
```

Here is the caller graph for this function:

8.18.3 Member Function Documentation

8.18.3.1 getInstance()

```
static KeyHandler main.KeyHandler.getInstance ( )  [static]
```

Gets the instance of the KeyHandler using the singleton pattern.

Returns

The KeyHandler instance.

Definition at line 30 of file KeyHandler.java.

```
00030                                                {
00031         if (instance == null) {
00032             instance = new KeyHandler();
00033         }
00034         return instance;
00035     }
```

Here is the call graph for this function:

8.18.3.2 keyPressed()

```
void main.KeyHandler.keyPressed (
             KeyEvent e )
```

Definition at line 43 of file KeyHandler.java.

```
00043                                            {
00044         int code = e.getKeyCode();
00045
00046         if (code == KeyEvent.VK_Z) {
00047             upPressed = true;
00048         }
00049         if (code == KeyEvent.VK_Q) {
00050             leftPressed = true;
00051         }
00052         if (code == KeyEvent.VK_S) {
00053             downPressed = true;
00054         }
00055         if (code == KeyEvent.VK_D) {
00056             rightPressed = true;
00057         }
00058         if (code == KeyEvent.VK_SPACE) {
00059             interactPressed = true;
00060         }
00061         if (code == KeyEvent.VK_ESCAPE) {
00062             escPressed = true;
00063         }
00064     }
```

8.18.3.3 keyReleased()

```
void main.KeyHandler.keyReleased (
             KeyEvent e )
```

Definition at line 67 of file KeyHandler.java.

```
00067                                             {
00068         int code = e.getKeyCode();
00069
00070         if (code == KeyEvent.VK_Z) {
00071             upPressed = false;
00072         }
00073         if (code == KeyEvent.VK_Q) {
00074             leftPressed = false;
00075         }
```

```
00076         if (code == KeyEvent.VK_S) {
00077             downPressed = false;
00078         }
00079         if (code == KeyEvent.VK_D) {
00080             rightPressed = false;
00081         }
00082         if (code == KeyEvent.VK_SPACE) {
00083             interactPressed = false;
00084         }
00085         if (code == KeyEvent.VK_ESCAPE) {
00086             escPressed = false;
00087         }
00088     }
```

8.18.3.4 keyTyped()

```
void main.KeyHandler.keyTyped (
             KeyEvent e )
```

Definition at line 38 of file KeyHandler.java.

```
00038                                     {
00039         // Not used
00040     }
```

8.18.4 Member Data Documentation

8.18.4.1 downPressed

```
boolean main.KeyHandler.downPressed
```

Definition at line 18 of file KeyHandler.java.

8.18.4.2 escPressed

```
boolean main.KeyHandler.escPressed
```

Definition at line 20 of file KeyHandler.java.

8.18.4.3 instance

```
KeyHandler main.KeyHandler.instance  [static]
```

Definition at line 17 of file KeyHandler.java.

8.18.4.4 interactPressed

```
boolean main.KeyHandler.interactPressed
```

Definition at line 19 of file KeyHandler.java.

8.18.4.5 leftPressed

```
boolean main.KeyHandler.leftPressed
```

Definition at line 18 of file KeyHandler.java.

8.18.4.6 rightPressed

```
boolean main.KeyHandler.rightPressed
```

Definition at line 18 of file KeyHandler.java.

8.18.4.7 upPressed

```
boolean main.KeyHandler.upPressed
```

Definition at line 18 of file KeyHandler.java.

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

- src/main/KeyHandler.java

8.19 item.armor.Legs Class Reference

Inheritance diagram for item.armor.Legs:

Collaboration diagram for item.armor.Legs:

Public Member Functions

- Legs ()

8.19.1 Detailed Description

Definition at line 3 of file Legs.java.

8.19.2 Constructor & Destructor Documentation

8.19.2.1 Legs()

```
item.armor.Legs.Legs ( )
```

Definition at line 5 of file Legs.java.

```
00005                 {
00006         System.out.println("Armure jambe");
00007     }
```

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

- src/item/armor/Legs.java

8.20 CreationCombatScene.Main Class Reference

Collaboration diagram for CreationCombatScene.Main:

8.21 main.Main Class Reference

Contains the main method to start the 2D Adventure game.

Collaboration diagram for main.Main:

Static Public Member Functions

- static void main (String[ ] args)
  The main method that initializes the game window and starts the game thread.

8.21.1 Detailed Description

Contains the main method to start the 2D Adventure game.

Definition at line 16 of file Main.java.

8.21.2 Member Function Documentation

8.21.2.1 main()

```
static void main.Main.main (
            String[] args )  [static]
```

The main method that initializes the game window and starts the game thread.

Parameters

args	Command-line arguments (not used in this application)

Definition at line 22 of file Main.java.

```
00022                                       {
00023         // Create a JFrame (window) for the game
00024         JFrame windows = new JFrame();
00025         windows.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
00026         windows.setResizable(false);
00027         windows.setTitle("2D Adventure");
00028
00029         // Create an instance of the Window class (game window)
00030         Window gameWindow = new Window();
00031         windows.add(gameWindow);
00032
00033         // Pack the components of the window
00034         windows.pack();
00035
00036         // Set the window to appear at the center of the screen
00037         windows.setLocationRelativeTo(null);
00038
00039         // Make the window visible
00040         windows.setVisible(true);
00041
00042         // Start the game thread to handle game logic and rendering
00043         gameWindow.startGameThread();
00044     }
```

Here is the call graph for this function:

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

- src/main/Main.java

8.22 item.potion.ManaPotion Class Reference

Inheritance diagram for item.potion.ManaPotion:

Collaboration diagram for item.potion.ManaPotion:

Public Member Functions

- ManaPotion ()

8.22.1 Detailed Description

Definition at line 3 of file ManaPotion.java.

8.22.2 Constructor & Destructor Documentation

8.22.2.1 ManaPotion()

```
item.potion.ManaPotion.ManaPotion ( )
```

Definition at line 5 of file ManaPotion.java.

```
00005                         {
00006         System.out.println("Potion de Mana");
00007     }
```

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

- src/item/potion/ManaPotion.java

8.23 UI.HUD.MenuType Enum Reference

Collaboration diagram for UI.HUD.MenuType:

Public Attributes

- WELCOME
- PAUSE
- FIGHT
- SHOP

8.23.1 Detailed Description

Definition at line 7 of file HUD.java.

8.23.2 Member Data Documentation

8.23.2.1 FIGHT

```
UI.HUD.MenuType.FIGHT
```

Definition at line 7 of file HUD.java.

8.23.2.2 PAUSE

```
UI.HUD.MenuType.PAUSE
```

Definition at line 7 of file HUD.java.

8.23.2.3 SHOP

`UI.HUD.MenuType.SHOP`

Definition at line 7 of file HUD.java.

8.23.2.4 WELCOME

`UI.HUD.MenuType.WELCOME`

Definition at line 7 of file HUD.java.

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

- src/UI/HUD.java

8.24 entity.props.OBJ_Chest Class Reference

Represents a chest object.

Inheritance diagram for entity.props.OBJ_Chest:

Collaboration diagram for entity.props.OBJ_Chest:

Public Member Functions

- OBJ_Chest (int worldX, int worldY)
  Constructor for the OBJ_Chest class.
- void playerInterraction (Player p)

Additional Inherited Members

8.24.1 Detailed Description

Represents a chest object.

Definition at line 18 of file OBJ_Chest.java.

8.24.2 Constructor & Destructor Documentation

8.24.2.1 OBJ_Chest()

```
entity.props.OBJ_Chest.OBJ_Chest (
            int worldX,
            int worldY )
```

Constructor for the OBJ_Chest class.

Initializes the name and loads the image for the chest.

Definition at line 24 of file OBJ_Chest.java.

```
00024                                           {
00025         super(worldX,worldY,"chest",1,0,true);
00026         loadTextures("chest");
00027         collision = true;
00028     }
```

Here is the call graph for this function:

8.24.3 Member Function Documentation

8.24.3.1 playerInterraction()

```
void entity.props.OBJ_Chest.playerInterraction (
            Player p )
```

Reimplemented from entity.Entity.

Definition at line 29 of file OBJ_Chest.java.

```
00029                                               {
00030         Item item= Generator.generateItem();
00031         if(true){
00032             p.addItem(item);
00033         }
00034         System.out.println("Chest interaction");
00035
00036         destroySelf();
00037
00038     }
```

Here is the call graph for this function:

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

- src/entity/props/OBJ_Chest.java

8.25 entity.props.OBJ_Door Class Reference

Represents a door object in the game.

Inheritance diagram for entity.props.OBJ_Door:

Collaboration diagram for entity.props.OBJ_Door:

Public Member Functions

- OBJ_Door (int worldX, int worldY)

  Constructor for the OBJ_Door class.
- void playerInterraction (Player p)

Additional Inherited Members

8.25.1 Detailed Description

Represents a door object in the game.

Definition at line 16 of file OBJ_Door.java.

8.25.2 Constructor & Destructor Documentation

8.25.2.1 OBJ_Door()

```
entity.props.OBJ_Door.OBJ_Door (
            int worldX,
            int worldY )
```

Constructor for the OBJ_Door class.

Initializes the name of the door and loads its image from a resource file.

Definition at line 22 of file OBJ_Door.java.

```
00022                                           {
00023         super(worldX, worldY,"door",1,0, true);
00024
00025         loadTextures("door");
00026         collision = true;
00027     }
```

Here is the call graph for this function:

8.25.3 Member Function Documentation

8.25.3.1 playerInterraction()

```
void entity.props.OBJ_Door.playerInterraction (
            Player p )
```

Reimplemented from entity.Entity.

Definition at line 28 of file OBJ_Door.java.

```
00028                                           {
00029         if(p.hasKey > 0 ) {
00030             // TODO : Change door collision; it's still here but with different textures and
      properties
00031             p.hasKey--;
00032 
00033             destroySelf();
00034         }
00035 
00036         System.out.println("Key:"+p.hasKey);
00037 
00038 
00039 
00040 
00041     }
```

Here is the call graph for this function:

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

- src/entity/props/OBJ_Door.java

8.26 entity.props.OBJ_Key Class Reference

Represents a key object in the game.

Inheritance diagram for entity.props.OBJ_Key:

Collaboration diagram for entity.props.OBJ_Key:

Public Member Functions

- OBJ_Key (int worldX, int worldY)
  Constructor for the OBJ_Key class.
- void playerInterraction (Player p)

Additional Inherited Members

8.26.1 Detailed Description

Represents a key object in the game.

Definition at line 16 of file OBJ_Key.java.

8.26.2 Constructor & Destructor Documentation

8.26.2.1 OBJ_Key()

```
entity.props.OBJ_Key.OBJ_Key (
             int worldX,
             int worldY )
```

Constructor for the OBJ_Key class.

Initializes the name of the key and loads its image from a file.

Definition at line 22 of file OBJ_Key.java.

```
00022                                           {
00023         super(worldX, worldY,"key",1,0,false);
00024
00025         loadTextures("key");
00026     }
```

Here is the call graph for this function:

8.26.3 Member Function Documentation

8.26.3.1 playerInterraction()

```
void entity.props.OBJ_Key.playerInterraction (
             Player p )
```

Reimplemented from entity.Entity.

Definition at line 27 of file OBJ_Key.java.

```
00027                                               {
00028
00029         p.hasKey++;
00030         System.out.println("Key:"+p.hasKey);
00031
00032         destroySelf();
00033     }
```

Here is the call graph for this function:

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

- src/entity/props/OBJ_Key.java

8.27 entity.Player Class Reference

Represents the player entity in the game.

Inheritance diagram for entity.Player:

Collaboration diagram for entity.Player:

Public Member Functions

- Player (String entityName, int worldX, int worldY, int dirX, int dirY, int speed, String facing, int spriteCntMax, int spriteSpeed)

  Constructor for the Player class.
- void update (Scene scene, double dt)

  Updates the player entity based on the current scene and time elapsed.
- Point checkObject (Entity entity, World world)

  Checks for collision with nearby objects using the player's hitbox.
- void pickUpObject (World gp, Point index)
- void addItem (Item i)

Public Attributes

- int hasKey = 0

Additional Inherited Members

8.27.1 Detailed Description

Represents the player entity in the game.

Definition at line 21 of file Player.java.

8.27.2 Constructor & Destructor Documentation

8.27.2.1 Player()

```
entity.Player.Player (
            String entityName,
            int worldX,
            int worldY,
            int dirX,
            int dirY,
            int speed,
            String facing,
            int spriteCntMax,
            int spriteSpeed )
```

Constructor for the Player class.

Parameters

worldX	The X-coordinate of the player in the world.
worldY	The Y-coordinate of the player in the world.
dirX	The X-direction of the player.
dirY	The Y-direction of the player.
speed	The speed of the player's movement.
facing	The direction the player is facing (up, down, left, right).
spriteCntMax	The maximum number of sprites for animation.
spriteSpeed	The speed of sprite animation.

Definition at line 35 of file Player.java.

```
00035
    {
00036         super(entityName, worldX, worldY, dirX, dirY, speed, facing, spriteCntMax, spriteSpeed);  //
    Calls the parent class for entity setup, specifying scene.keyH for player
00037     }
```

8.27.3 Member Function Documentation

8.27.3.1 addItem()

```
void entity.Player.addItem (
             Item i )
```

Definition at line 124 of file Player.java.

```
00124                                {
00125         System.out.println("J'ai rajoutée un item");
00126     }
```

Here is the caller graph for this function:

8.27.3.2 checkObject()

```
Point entity.Player.checkObject (
             Entity entity,
             World world )
```

Checks for collision with nearby objects using the player's hitbox.

Parameters

entity	The entity to check collision for.
world	The current game world.

Returns

The coordinates of the collided object or null if no collision.

Definition at line 95 of file Player.java.

```
00095                                                    {
00096
00097         Point index = null;
00098
00099         for (Props obj :  world.objMap.values()) {
00100             if (obj != null) {
00101                 if (entity.hitbox.intersects(obj.hitbox)) {
00102                     if (obj.getCollision()) {   //If object has "solid" collision
00103                         //prevent the player from moving in the hitbox
00104
00105                     }
00106                     index = new Point((int) obj.worldX, (int) obj.worldY);
00107                     break;
00108                 }
00109             }
00110         }
00111
00112         return index;
00113     }
```

Here is the caller graph for this function:

8.27.3.3 pickUpObject()

```
void entity.Player.pickUpObject (
            World gp,
            Point index )
```

Definition at line 117 of file Player.java.

```
00117                                                   {
00118         if (index != null) {
00119             Entity object = gp.objMap.get(index);
00120             object.playerInterraction(this);
00121         }
00122     }
```

Here is the caller graph for this function:

8.27.3.4 update()

```
void entity.Player.update (
            Scene scene,
            double dt )
```

Updates the player entity based on the current scene and time elapsed.

Parameters

scene	The current game scene.
dt	The time elapsed since the last update.

Reimplemented from entity.Character.

Definition at line 44 of file Player.java.

```
00044                                           {
00045         super.update(scene, dt);  // Calls the parent class update method
00046         // World updates
00047         if (scene.state == State.WORLD) {
00048             if (scene.keyH.upPressed || scene.keyH.downPressed || scene.keyH.leftPressed ||
      scene.keyH.rightPressed) {
00049                 dirX = 0;
00050                 dirY = 0;
00051                 if (scene.keyH.leftPressed) {
00052                     dirX = -1;
00053                     facing = "left";
00054                 }
00055                 if (scene.keyH.rightPressed) {
00056                     dirX = 1;
00057                     facing = "right";
00058                 }
00059                 if (scene.keyH.upPressed) {
00060                     dirY = -1;
00061                     facing = "up";
00062                 }
00063                 if (scene.keyH.downPressed) {
00064                     dirY = 1;
00065                     facing = "down";
00066                 }
00067                 accelerate(30,20, dt);
00068             } else {
00069                 if (speed > 0) {
00070                     decelerate(1,dt);
00071                 }
00072             }
00073
00074             // CHECK OBJECT COLLISION
00075             Point objIndex = checkObject(this, World.getWorld());
00076             pickUpObject(World.getWorld(), objIndex);
00077
00078
```

```
00079         }
00080
00081         // Fightscene updates
00082         if (scene.state == State.FIGHT) {
00083             // To be implemented
00084
00085
00086         }
00087     }
```

Here is the call graph for this function: Here is the caller graph for this function:

8.27.4 Member Data Documentation

8.27.4.1 hasKey

```
int entity.Player.hasKey = 0
```

Definition at line 23 of file Player.java.

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

- src/entity/Player.java

8.28 CreationCombatScene.entity.PlayerTest Class Reference

Inheritance diagram for CreationCombatScene.entity.PlayerTest:

Collaboration diagram for CreationCombatScene.entity.PlayerTest:

Public Member Functions

- PlayerTest ()
- void update (double dt)

Additional Inherited Members

8.28.1 Detailed Description

Definition at line 5 of file PlayerTest.java.

8.28.2 Constructor & Destructor Documentation

8.28.2.1 PlayerTest()

```
CreationCombatScene.entity.PlayerTest.PlayerTest ( )
```

Definition at line 8 of file PlayerTest.java.

```
00008                           {
00009         super(0,0,0,0,0," ",0,0);
00010     }
```

8.28.3 Member Function Documentation

8.28.3.1 update()

```
void CreationCombatScene.entity.PlayerTest.update (
             double dt )
```

Definition at line 12 of file PlayerTest.java.

```
00012                                   {
00013
00014         System.out.println("Je suis modifié");
00015
00016     }
```

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

- test/CreationCombatScene/entity/PlayerTest.java

8.29 item.potion.Potion Class Reference

Inheritance diagram for item.potion.Potion:

Collaboration diagram for item.potion.Potion:

8.29.1 Detailed Description

Definition at line 5 of file Potion.java.

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

- src/item/potion/Potion.java

8.30 entity.props.Props Class Reference

Represents in-game props with properties such as image, name, and position.

Inheritance diagram for entity.props.Props:

Collaboration diagram for entity.props.Props:

Public Member Functions

- boolean getCollision ()
- void draw (Graphics2D g2, World world)

  Draw the prop on the screen based on its position relative to the player's position.
- void destroySelf ()

Protected Member Functions

- void loadTextures (String name)

Package Functions

- Props (int x, int y, String name, int spriteCntMax, int spriteSpeed, boolean collision)

Private Attributes

- BufferedImage image

Additional Inherited Members

8.30.1 Detailed Description

Represents in-game props with properties such as image, name, and position.

Definition at line 24 of file Props.java.

8.30.2 Constructor & Destructor Documentation

8.30.2.1 Props()

```
entity.props.Props.Props (
            int x,
            int y,
            String name,
            int spriteCntMax,
            int spriteSpeed,
            boolean collision )  [package]
```

Definition at line 28 of file Props.java.

```
00028                                                                                {
00029         super(name,x,y,spriteCntMax,spriteSpeed,collision);
00030         hitbox.width = 3*Const.WRLD_tileScreenSize/4;  //Slightly smaller than a tile
00031         hitbox.height = 3*Const.WRLD_tileScreenSize/4;
00032         hitbox.x = worldX + hitbox.width/4;
00033         hitbox.y = worldY + hitbox.height/4;
00034     }
```

8.30.3 Member Function Documentation

8.30.3.1 destroySelf()

```
void entity.props.Props.destroySelf ( )
```

Definition at line 77 of file Props.java.

```
00077                               {
00078         Point point=new Point(worldX,worldY);
00079         World instance=World.getWorld();
00080         instance.objMap.remove(point,this);
00081     }
```

Here is the call graph for this function: Here is the caller graph for this function:

8.30.3.2 draw()

```
void entity.props.Props.draw (
             Graphics2D g2,
             World world )
```

Draw the prop on the screen based on its position relative to the player's position.

Parameters

g2	Graphics2D object for drawing.
world	World object containing information about the game world.

Definition at line 57 of file Props.java.

```
00057                                                {
00058         // Calculate the screen position of the player
00059         int playerScreenX = (Const.WDW_width - Const.WRLD_entityScreenSize) / 2;
00060         int playerScreenY = (Const.WDW_height - Const.WRLD_entityScreenSize) / 2;
00061
00062         // Calculate the screen position of the prop relative to the player's position
00063         int screenX = worldX - world.player.worldX + playerScreenX;
00064         int screenY = worldY - world.player.worldY + playerScreenY;
00065
00066         // Check if the prop is within the visible screen region around the player
00067         if (worldX + Const.WRLD_tileScreenSize > world.player.worldX - playerScreenX
00068                 && worldX - Const.WRLD_tileScreenSize < world.player.worldX + playerScreenX
00069                 && worldY + Const.WRLD_tileScreenSize > world.player.worldY - playerScreenY
00070                 && worldY - Const.WRLD_tileScreenSize < world.player.worldY + playerScreenY) {
00071             // Draw the prop on the screen
00072             g2.drawImage(image, screenX, screenY, Const.WRLD_entityScreenSize,
      Const.WRLD_entityScreenSize, null);
00073             g2.drawRect(screenX + Const.WRLD_entityScreenSize/8, screenY +
      Const.WRLD_entityScreenSize/8, this.hitbox.width, this.hitbox.height);
00074         }
00075     }
```

8.30.3.3 getCollision()

```
boolean entity.props.Props.getCollision ( )
```

Definition at line 36 of file Props.java.

```
00036                                   {
00037         return collision;
00038     }
```

8.30.3.4 loadTextures()

```
void entity.props.Props.loadTextures (
             String name )  [protected]
```

Definition at line 40 of file Props.java.

```
00040                                           {
00041         try {
00042             // Load the image of the key from the specified file path
00043             image = ImageIO.read(new FileInputStream("res/object/"+name+".png"));
00044         } catch (IOException e) {
00045             // Print the stack trace in case of an IOException during image loading
00046             e.printStackTrace();
00047         }
00048     }
```

8.30.4 Member Data Documentation

8.30.4.1 image

```
BufferedImage entity.props.Props.image  [private]
```

Definition at line 25 of file Props.java.

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

- src/entity/props/Props.java

8.31 game.Scene Class Reference

Represents an abstract scene in the game.

Inheritance diagram for game.Scene:

Collaboration diagram for game.Scene:

Classes

- enum State

Public Member Functions

- abstract void update ()

  Updates the scene.
- abstract void draw (Graphics2D g2, int screenWidth, int screenHeight)

  Draws the scene.
- void checkPauseScene ()

  Gets the World scene.
- void changeScene (State newState)

Static Public Member Functions

- static double getdt ()

Public Attributes

- KeyHandler keyH = KeyHandler.getInstance()
- State state
- HUD menu

Static Protected Attributes

- static double dt = 0

Private Attributes

- State _lastState

8.31.1 Detailed Description

Represents an abstract scene in the game.

Definition at line 19 of file Scene.java.

8.31.2 Member Function Documentation

8.31.2.1 changeScene()

```
void game.Scene.changeScene (
            State newState )
```

Definition at line 91 of file Scene.java.

```
00091                                           {
00092         System.out.println("CHANGING SCENE TO: " + newState);
00093         this.state = newState;
00094     }
```

8.31.2.2 checkPauseScene()

```
void game.Scene.checkPauseScene ( )
```

Gets the World scene.

Returns

The World scene.

Creates and returns a new FightScene.

Parameters

player	The player entity in the fight.
enemy	The enemy entity in the fight.

Returns

The new FightScene.

Checks for a change in the scene state based on user input.

Definition at line 74 of file Scene.java.

```
00074                                   {
00075         if (keyH.escPressed) {
00076             keyH.escPressed = false;
00077
00078             if (state != State.PAUSE) {
00079                 System.out.println("CHANGING SCENE TO: PAUSE");
00080                 _lastState = state;
00081                 state = State.PAUSE;
00082                 menu = new HUD(MenuType.PAUSE);
00083             } else {
00084                 System.out.println("CHANGING SCENE TO: "+_lastState);
00085                 state = _lastState;
00086                 menu = null;
00087             }
00088         }
00089     }
```

Here is the caller graph for this function:

8.31.2.3 draw()

```
abstract void game.Scene.draw (
            Graphics2D g2,
            int screenWidth,
            int screenHeight )  [abstract]
```

Draws the scene.

Parameters

g2	The Graphics2D object for drawing.
screenWidth	The width of the screen.
screenHeight	The height of the screen.

Reimplemented in game.World.

Here is the caller graph for this function:

8.31.2.4 getdt()

```
static double game.Scene.getdt ( )  [static]
```

Definition at line 48 of file Scene.java.

```
00048                                {
00049         return dt;
00050     }
```

8.31.2.5 update()

```
abstract void game.Scene.update ( )  [abstract]
```

Updates the scene.

Reimplemented in game.World.

Here is the caller graph for this function:

8.31.3 Member Data Documentation

8.31.3.1 _lastState

```
State game.Scene._lastState  [private]
```

Definition at line 27 of file Scene.java.

8.31.3.2 dt

```
double game.Scene.dt = 0  [static], [protected]
```

Definition at line 30 of file Scene.java.

8.31.3.3 keyH

```
KeyHandler game.Scene.keyH = KeyHandler.getInstance()
```

Definition at line 29 of file Scene.java.

8.31.3.4 menu

```
HUD game.Scene.menu
```

Definition at line 32 of file Scene.java.

8.31.3.5 state

```
State game.Scene.state
```

Definition at line 31 of file Scene.java.

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

- src/game/Scene.java

8.32 item.potion.SpeedPotion Class Reference

Inheritance diagram for item.potion.SpeedPotion:

Collaboration diagram for item.potion.SpeedPotion:

Public Member Functions

- SpeedPotion ()

8.32.1 Detailed Description

Definition at line 3 of file SpeedPotion.java.

8.32.2 Constructor & Destructor Documentation

8.32.2.1 SpeedPotion()

```
item.potion.SpeedPotion.SpeedPotion ( )
```

Definition at line 5 of file SpeedPotion.java.

```
00005                           {
00006         System.out.println("Potion de vitesse");
00007     }
```

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

- src/item/potion/SpeedPotion.java

8.33 item.weapon.Staff Class Reference

Inheritance diagram for item.weapon.Staff:

Collaboration diagram for item.weapon.Staff:

Public Member Functions

- Staff ()

8.33.1 Detailed Description

Definition at line 3 of file Staff.java.

8.33.2 Constructor & Destructor Documentation

8.33.2.1 Staff()

```
item.weapon.Staff.Staff ( )
```

Definition at line 5 of file Staff.java.

```
00005                   {
00006         System.out.println("Staff");
00007     }
```

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

- src/item/weapon/Staff.java

8.34 game.Scene.State Enum Reference

Collaboration diagram for game.Scene.State:

Public Attributes

- WORLD
- FIGHT
- PAUSE
- MENU

8.34.1 Detailed Description

Definition at line 25 of file Scene.java.

8.34.2 Member Data Documentation

8.34.2.1 FIGHT

`game.Scene.State.FIGHT`

Definition at line 25 of file Scene.java.

8.34.2.2 MENU

`game.Scene.State.MENU`

Definition at line 25 of file Scene.java.

8.34.2.3 PAUSE

`game.Scene.State.PAUSE`

Definition at line 25 of file Scene.java.

8.34.2.4 WORLD

`game.Scene.State.WORLD`

Definition at line 25 of file Scene.java.

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

- src/game/Scene.java

8.35 item.weapon.Sword Class Reference

Inheritance diagram for item.weapon.Sword:

Collaboration diagram for item.weapon.Sword:

Public Member Functions

- Sword ()

8.35.1 Detailed Description

Definition at line 3 of file Sword.java.

8.35.2 Constructor & Destructor Documentation

8.35.2.1 Sword()

```
item.weapon.Sword.Sword ( )
```

Definition at line 5 of file Sword.java.

```
00005                     {
00006         System.out.println("Armure foot");
00007     }
```

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

- src/item/weapon/Sword.java

8.36 UI.Textbox Class Reference

Inheritance diagram for UI.Textbox:

Collaboration diagram for UI.Textbox:

Public Member Functions

- Textbox (String text, String fontName, int w, int h, Color color)
- void draw (Graphics2D g2, int x, int y)

Static Public Member Functions

- static void loadFont (String fontName)

Private Attributes

- int _width
- int _height
- Color _color
- String _text
- Font _font
- int _fontSizeToUse

8.36.1 Detailed Description

Definition at line 12 of file Textbox.java.

8.36.2 Constructor & Destructor Documentation

8.36.2.1 Textbox()

```
UI.Textbox.Textbox (
            String text,
            String fontName,
            int w,
            int h,
            Color color )
```

Definition at line 20 of file Textbox.java.

```
00020                                                                    {
00021         _width = w; _height = h;
00022         _text = text;
00023         _color = color;
00024
00025         //loadFont(fontName);
00026         _font = new Font(fontName, Font.PLAIN, 1);
00027
00028
00029         int stringWidth = this.getFontMetrics(_font).stringWidth(_text);
00030         int componentWidth = _width;
00031
00032         // Find out how much the font can grow in width.
00033         double widthRatio = (double)componentWidth / (double)stringWidth;
00034
00035         int newFontSize = (int)(_font.getSize() * widthRatio);
00036         int componentHeight = _height;
00037
00038         // Pick a new font size so it will not be larger than the height of label.
00039         _fontSizeToUse = Math.min(newFontSize, componentHeight);
00040
00041         // Set the label's font size to the newly determined size.
00042         _font = new Font(fontName, Font.PLAIN, _fontSizeToUse);
00043     }
```

8.36.3 Member Function Documentation

8.36.3.1 draw()

```
void UI.Textbox.draw (
            Graphics2D g2,
            int x,
            int y )
```

Definition at line 59 of file Textbox.java.

```
00059                                                {
00060         //TODO: find a way to center text on the button
00061         g2.setFont(_font);
00062         g2.setColor(Color.black);
00063         g2.drawRect(x,y,_width,_height);
00064         g2.setColor(_color);
00065         g2.drawString(_text, x + _fontSizeToUse, y + (_height +_fontSizeToUse)/2);
00066     }
```

Here is the caller graph for this function:

8.36.3.2 loadFont()

```
static void UI.Textbox.loadFont (
            String fontName )  [static]
```

Definition at line 46 of file Textbox.java.

```
00046                                                {
00047         try {
00048             GraphicsEnvironment ge = GraphicsEnvironment.getLocalGraphicsEnvironment();
00049             ge.registerFont(Font.createFont(Font.TRUETYPE_FONT, new
      File("res/hud/font/rainyhearts.ttf")));
00050         }
00051         catch (FontFormatException e) {
00052             e.printStackTrace();
00053         }
00054         catch (IOException e) {
00055             e.printStackTrace();
00056         }
00057     }
```

Here is the caller graph for this function:

8.36.4 Member Data Documentation

8.36.4.1 _color

```
Color UI.Textbox._color  [private]
```

Definition at line 14 of file Textbox.java.

8.36.4.2 _font

```
Font UI.Textbox._font  [private]
```

Definition at line 16 of file Textbox.java.

8.36.4.3 _fontSizeToUse

```
int UI.Textbox._fontSizeToUse  [private]
```

Definition at line 17 of file Textbox.java.

8.36.4.4 _height

```
int UI.Textbox._height  [private]
```

Definition at line 13 of file Textbox.java.

8.36.4.5 _text

```
String UI.Textbox._text  [private]
```

Definition at line 15 of file Textbox.java.

8.36.4.6 _width

```
int UI.Textbox._width  [private]
```

Definition at line 13 of file Textbox.java.

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

- src/UI/Textbox.java

8.37 tiles.Tile Class Reference

Represents a tile in the game world.

Collaboration diagram for tiles.Tile:

Public Member Functions

- Tile (int spriteCntMax, int spriteSpeed, boolean isBlocking, int ind)

  Constructor for the Tile class.
- void setCollision (boolean collision)

  Set whether the tile is blocking.
- void setTexture (BufferedImage[ ] newTexture)

  Set the texture for the tile.
- void setPos (int x, int y)

  Set the position of the tile.
- int[ ] getPos ()

  Get the position of the tile.
- boolean getCollision ()

  Get whether the tile is blocking.
- void setSpriteCountAndSpeed (int newSpriteCntMax, int newSpriteSpeed)

  Set the sprite count and speed for the tile.
- void loadTextures (String name, boolean animated, int i)

  Load textures for the tile.
- void updateFrames ()

  Update the sprite frames for the tile.
- void draw (Graphics2D g2, int screenX, int screenY)

  Draw the tile on the screen.

Public Attributes

- final int screenSize = tileSize ∗ scale
- int spriteSpeed
- int spriteCntMax
- BufferedImage[ ] image

Private Attributes

- boolean _isBlocking = false
- int _worldX
- int _worldY
- final int tileSize = 16
- final int scale = 3
- int _spriteCnt = 0
- int _spriteUpdater = 0

8.37.1 Detailed Description

Represents a tile in the game world.

Definition at line 21 of file Tile.java.

8.37.2 Constructor & Destructor Documentation

8.37.2.1 Tile()

```
tiles.Tile.Tile (
            int spriteCntMax,
            int spriteSpeed,
            boolean isBlocking,
            int ind )
```

Constructor for the Tile class.

Parameters

spriteCntMax	The maximum number of sprites for the tile.
spriteSpeed	The speed of sprite animation.
isBlocking	Indicates whether the tile is blocking.
ind	The index name of the tile.

Definition at line 46 of file Tile.java.

```
00046                                                                                {
00047         this.spriteCntMax = spriteCntMax;
00048         this.spriteSpeed = spriteSpeed;
00049         this._isBlocking = isBlocking;
00050         image = new BufferedImage[spriteCntMax];
00051     }
```

8.37.3 Member Function Documentation

8.37.3.1 draw()

```
void tiles.Tile.draw (
            Graphics2D g2,
            int screenX,
            int screenY )
```

Draw the tile on the screen.

Parameters

g2	Graphics2D object for drawing.
screenX	The x-coordinate on the screen.
screenY	The y-coordinate on the screen.

Definition at line 158 of file Tile.java.

```
00158                                                                    {
00159         BufferedImage render = null;
00160
00161         for (int i = 0; i < spriteCntMax; i++) {
00162             if (_spriteCnt == i) {
00163                 render = image[i];
00164             }
00165         }
00166
00167         g2.drawImage(render, screenX, screenY, screenSize, screenSize, null);
```

```
00168         //g2.drawRect(screenX,screenY,Const.WRLD_tileScreenSize,Const.WRLD_tileScreenSize);
      //Debugging purposes
00169     }
```

Here is the caller graph for this function:

8.37.3.2 getCollision()

```
boolean tiles.Tile.getCollision ( )
```

Get whether the tile is blocking.

Returns

True if the tile is blocking, false otherwise.

Definition at line 99 of file Tile.java.

```
00099                                {
00100         return _isBlocking;
00101     }
```

Here is the caller graph for this function:

8.37.3.3 getPos()

```
int[] tiles.Tile.getPos ( )
```

Get the position of the tile.

Returns

An array containing the x and y coordinates of the tile.

Definition at line 87 of file Tile.java.

```
00087                               {
00088         int tmp[] = new int[2];
00089         tmp[0] = _worldX;
00090         tmp[1] = _worldY;
00091         return tmp;
00092     }
```

Here is the caller graph for this function:

8.37.3.4 loadTextures()

```
void tiles.Tile.loadTextures (
             String name,
             boolean animated,
             int i )
```

Load textures for the tile.

Parameters

name	The name of the tile.
animated	Indicates whether the tile has animated textures.
i	Index variable for static textures.

Definition at line 121 of file Tile.java.

```
00121                                                                   {
00122         try {
00123             if (animated) {
00124                 for (int j = 0; j < spriteCntMax; j++) {
00125                     image[j] = ImageIO.read(new FileInputStream("res/tiles/animated/" + name + (j + 1)
      + ".png"));
00126                 }
00127             } else {
00128                 image[0] = ImageIO.read(new FileInputStream("res/tiles/static/" + name + (i + 1) +
      ".png"));
00129             }
00130         } catch (IOException e) {
00131             e.printStackTrace();
00132         }
00133     }
```

8.37.3.5 setCollision()

```
void tiles.Tile.setCollision (
             boolean collision )
```

Set whether the tile is blocking.

Parameters

collision	Indicates whether the tile is blocking.

Definition at line 58 of file Tile.java.

```
00058                                                {
00059         _isBlocking = collision;
00060     }
```

Here is the caller graph for this function:

8.37.3.6 setPos()

```
void tiles.Tile.setPos (
             int x,
             int y )
```

Set the position of the tile.

Parameters

x	The x-coordinate of the tile.
y	The y-coordinate of the tile.

Definition at line 77 of file Tile.java.

```
00077                                       {
00078         this._worldX = x;
00079         this._worldY = y;
00080     }
```

Here is the caller graph for this function:

8.37.3.7 setSpriteCountAndSpeed()

```
void tiles.Tile.setSpriteCountAndSpeed (
             int newSpriteCntMax,
             int newSpriteSpeed )
```

Set the sprite count and speed for the tile.

Parameters

newSpriteCntMax	The new maximum number of sprites for the tile.
newSpriteSpeed	The new speed of sprite animation for the tile.

Definition at line 109 of file Tile.java.

```
00109                                                                                   {
00110         this.spriteCntMax = newSpriteCntMax;
00111         this.spriteSpeed = newSpriteSpeed;
00112     }
```

8.37.3.8 setTexture()

```
void tiles.Tile.setTexture (
             BufferedImage[] newTexture )
```

Set the texture for the tile.

Parameters

newTexture	The new texture for the tile.

Definition at line 67 of file Tile.java.

```
00067                                                   {
00068         this.image = newTexture;
00069     }
```

Here is the caller graph for this function:

8.37.3.9 updateFrames()

```
void tiles.Tile.updateFrames ( )
```

Update the sprite frames for the tile.

Definition at line 138 of file Tile.java.

```
00138                                 {
00139         if (spriteSpeed > 0) {
00140             _spriteUpdater++;
00141             if (_spriteUpdater > spriteSpeed) {
00142                 _spriteCnt++;
00143                 if (_spriteCnt == spriteCntMax) {
00144                     _spriteCnt = 0;
00145                 }
00146                 _spriteUpdater = 1;
00147             }
00148         }
00149     }
```

Here is the caller graph for this function:

8.37.4 Member Data Documentation

8.37.4.1 _isBlocking

```
boolean tiles.Tile._isBlocking = false  [private]
```

Definition at line 22 of file Tile.java.

8.37.4.2 _spriteCnt

```
int tiles.Tile._spriteCnt = 0  [private]
```

Definition at line 30 of file Tile.java.

8.37.4.3 _spriteUpdater

```
int tiles.Tile._spriteUpdater = 0  [private]
```

Definition at line 31 of file Tile.java.

8.37.4.4 _worldX

```
int tiles.Tile._worldX  [private]
```

Definition at line 23 of file Tile.java.

8.37.4.5 _worldY

```
int tiles.Tile._worldY  [private]
```

Definition at line 23 of file Tile.java.

8.37.4.6 image

```
BufferedImage [] tiles.Tile.image
```

Definition at line 36 of file Tile.java.

8.37.4.7 scale

```
final int tiles.Tile.scale = 3  [private]
```

Definition at line 27 of file Tile.java.

8.37.4.8 screenSize

```
final int tiles.Tile.screenSize = tileSize * scale
```

Definition at line 28 of file Tile.java.

8.37.4.9 spriteCntMax

```
int tiles.Tile.spriteCntMax
```

Definition at line 34 of file Tile.java.

8.37.4.10 spriteSpeed

```
int tiles.Tile.spriteSpeed
```

Definition at line 32 of file Tile.java.

8.37.4.11 tileSize

```
final int tiles.Tile.tileSize = 16  [private]
```

Definition at line 26 of file Tile.java.

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

- src/tiles/Tile.java

8.38 tiles.TileManager Class Reference

Manages tiles in the game world.

Collaboration diagram for tiles.TileManager:

Public Member Functions

- TileManager (World world)

  Constructor for the TileManager class.
- Tile getTile (int x, int y)

  Gets the tile of the top map with its x, y coordinates.
- void update (World world)

  Updates the tile frames based on the player's position.
- void draw (Graphics2D g2, World world)

  Draws the tiles in the game world.

Private Member Functions

- void storeTexture (String name, Tile[ ] tiles, int start, int size, boolean animated, int spriteCntMax, int spriteSpeed, boolean isBlocking)

  Stores in a Tile array (from "start" to "start + size") the textures that are loaded for the game.
- void loadTextures ()

  Loading all tiles' textures of the game for the map in an array.
- void loadMap (String filePath, World world, Tile[ ][ ] mapTile, Tile[ ] textures)

  Reads the txt map file to store the tileMap accordingly.

Private Attributes

- Tile[ ] _floorMapTextures
- Tile[ ] _topMapTextures
- Tile[ ][ ] _floorMap
- Tile[ ][ ] _topMap

8.38.1 Detailed Description

Manages tiles in the game world.

Definition at line 20 of file TileManager.java.

8.38.2 Constructor & Destructor Documentation

8.38.2.1 TileManager()

```
tiles.TileManager.TileManager (
            World world )
```

Constructor for the TileManager class.

Parameters

world	The world for which tiles are managed.

Definition at line 34 of file TileManager.java.

```
00034                                   {
00035         // Textures
00036         _floorMapTextures = new Tile[Const.nbFloorTextures];
00037         _topMapTextures = new Tile[Const.nbTopTextures];
00038 
00039         // Map itself
00040         _floorMap = new Tile[Const.WRLD_maxRow][Const.WRLD_maxCol];
00041         _topMap = new Tile[Const.WRLD_maxRow][Const.WRLD_maxCol];
00042 
00043         loadTextures();
00044         loadMap("res/maps/debugfloor.txt", world, _floorMap, _floorMapTextures);
00045         loadMap("res/maps/debugtop.txt", world, _topMap, _topMapTextures);
00046     }
```

Here is the call graph for this function:

8.38.3 Member Function Documentation

8.38.3.1 draw()

```
void tiles.TileManager.draw (
            Graphics2D g2,
            World world )
```

Draws the tiles in the game world.

Parameters

g2	The graphics context.
world	The world object.
screenWidth	The screen width.
screenHeight	The screen height.

Definition at line 185 of file TileManager.java.

```
00185                                                   {
00186         int playerScreenX = (Const.WDW_width - Const.WRLD_entityScreenSize) / 2;
00187         int playerScreenY = (Const.WDW_height - Const.WRLD_entityScreenSize) / 2;
00188 
00189         // Check for every tile of the map if it needs to be drawn
00190         for (int i = 0; i < Const.WRLD_maxRow; i++) {
00191             for (int j = 0; j < Const.WRLD_maxCol; j++) {
00192                 int worldX = _floorMap[i][j].getPos()[0];
00193                 int worldY = _floorMap[i][j].getPos()[1];
00194 
00195                 // Checks if the player is close enough to the tile to render it to optimize memory
      and CPU usage
00196                 if (worldX + Const.WRLD_tileScreenSize > world.player.worldX - playerScreenX
00197                         && worldX - Const.WRLD_tileScreenSize < world.player.worldX + playerScreenX
00198                         && worldY + Const.WRLD_tileScreenSize > world.player.worldY - playerScreenY
00199                         && worldY - Const.WRLD_tileScreenSize < world.player.worldY + playerScreenY) {
00200                     int screenX = worldX - world.player.worldX + playerScreenX;
00201                     int screenY = worldY - world.player.worldY + playerScreenY;
00202                     _floorMap[i][j].draw(g2, screenX, screenY);
00203                     _topMap[i][j].draw(g2, screenX, screenY);
00204                 }
00205             }
00206         }
00207     }
```

Here is the call graph for this function: Here is the caller graph for this function:

8.38.3.2 getTile()

```
Tile tiles.TileManager.getTile (
            int x,
            int y )
```

Gets the tile of the top map with its x, y coordinates.

Parameters

x	X-coordinate of the tile.
y	Y-coordinate of the tile.

Returns

The corresponding tile of the top map.

Definition at line 55 of file TileManager.java.

```
00055                                       {
00056         int param = Const.WRLD_tileScreenSize;
00057         int i = x / param;
00058         int j = y / param;
00059 
00060         return _topMap[j][i];
00061     }
```

Here is the caller graph for this function:

8.38.3.3 loadMap()

```
void tiles.TileManager.loadMap (
            String filePath,
            World world,
            Tile mapTile[][],
            Tile[] textures )  [private]
```

Reads the txt map file to store the tileMap accordingly.

Parameters

filePath	Path of the txt map.
world	World to get the size.
mapTile	The map in which we store the read tiles on the txt.
textures	The textures tile array where we read the textures from.

Definition at line 116 of file TileManager.java.

```
00116                                                                                                   {
00117         try {
00118             File file = new File(filePath);
00119             FileReader fileReader = new FileReader(file);
00120             BufferedReader br = new BufferedReader(fileReader);
00121 
00122             for (int i = 0; i < Const.WRLD_maxRow; i++) {
00123                 String line = br.readLine();
00124 
00125                 for (int j = 0; j < Const.WRLD_maxCol; j++) {
00126                     String[] numbers = line.split("\\s+");
00127                     int num = Integer.parseInt(numbers[j]); // Reading the file itself and stocking
      int read
00128 
00129                     Tile tileCurrent = new Tile(textures[num].spriteCntMax, textures[num].spriteSpeed,
00130                             textures[num].getCollision(), num); // Creating new tile to store with the
      according texture
00131                     tileCurrent.setPos(Const.WRLD_tileScreenSize * j, Const.WRLD_tileScreenSize * i);
00132 
00133                     mapTile[i][j] = tileCurrent; // Setting the tile to the actual map
00134                     mapTile[i][j].setTexture(textures[num].image); // Set the current tile texture to
      what corresponds
00135                     // in the textures array
00136                     mapTile[i][j].setCollision(textures[num].getCollision()); // Set the collision
      factor to the tile
00137 
00138                     // System.out.print(num + " ");
00139                 }
00140                 // System.out.println("");
00141             }
00142             br.close();
00143 
00144         } catch (Exception e) {
00145             e.printStackTrace();
00146         }
00147     }
```

Here is the call graph for this function: Here is the caller graph for this function:

8.38.3.4 loadTextures()

```
void tiles.TileManager.loadTextures ( )  [private]
```

Loading all tiles' textures of the game for the map in an array.

Definition at line 87 of file TileManager.java.

```
00087                                 {
00088         // Floor map textures
00089         storeTexture("grass", _floorMapTextures, 0, 3, false, 1, 0, false);
00090         // Decoration textures (plants and tall grass)
00091         storeTexture("grass0", _floorMapTextures, 3, 1, true, 5, 20, false);
00092         storeTexture("grass1", _floorMapTextures, 4, 1, true, 5, 20, false);
00093         storeTexture("flower0", _floorMapTextures, 5, 1, true, 6, 20, false);
00094 
00095         // Top map textures (trees & forest)
00096         storeTexture("void", _topMapTextures, 0, 1, false, 1, 0, false);
00097 
00098         storeTexture("forest", _topMapTextures, 1, 9, false, 1, 0, true);
00099         storeTexture("tree", _topMapTextures, 10, 9, false, 1, 0, true);
00100 
00101         storeTexture("forest_topleft", _topMapTextures, 19, 3, false, 1, 0, true);
00102         storeTexture("forest_topright", _topMapTextures, 22, 2, false, 1, 0, true);
00103         storeTexture("forest_bottomleft", _topMapTextures, 24, 2, false, 1, 0, true);
00104         storeTexture("forest_bottomright", _topMapTextures, 26, 1, false, 1, 0, true);
00105         storeTexture("fire", _topMapTextures, 27, 1, true, 7, 15, true);
00106     }
```

Here is the call graph for this function: Here is the caller graph for this function:

8.38.3.5 storeTexture()

```
void tiles.TileManager.storeTexture (
            String name,
            Tile[] tiles,
            int start,
            int size,
            boolean animated,
            int spriteCntMax,
            int spriteSpeed,
            boolean isBlocking )  [private]
```

Stores in a Tile array (from "start" to "start + size") the textures that are loaded for the game.

Parameters

name	Name of the texture.
tiles	Array in which to store the textures.
start	Starting point of loading textures for the array.
size	If there are multiple textures with the same name but with variations (ex : grass1, grass2...).
animated	Boolean if the textures are supposed to be animated (different folder if animated or not).
spriteCntMax	Number of sprites if the texture is animated (if not, then put 1).
spriteSpeed	Sprite speed if the texture is animated (if not, then put 0).
isBlocking	Boolean to make the player collide with the tile or not.

Definition at line 75 of file TileManager.java.

```
00076                                                                   {
00077         for (int i = start; i < size + start; i++) {
00078             Tile tile = new Tile(spriteCntMax, spriteSpeed, isBlocking, i);
00079             tiles[i] = tile;
00080             tiles[i].loadTextures(name, animated, i - start);
00081         }
00082     }
```

Here is the caller graph for this function:

8.38.3.6 update()

```
void tiles.TileManager.update (
            World world )
```

Updates the tile frames based on the player's position.

Parameters

world	The world object.
screenWidth	The screen width.
screenHeight	The screen height.

Definition at line 156 of file TileManager.java.

```
00156                                   {
00157         int playerScreenX = (Const.WDW_width - Const.WRLD_entityScreenSize) / 2;
00158         int playerScreenY = (Const.WDW_height - Const.WRLD_entityScreenSize) / 2;
00159
00160         for (int i = 0; i < Const.WRLD_maxRow; i++) {
```

```
00161             for (int j = 0; j < Const.WRLD_maxCol; j++) {
00162                 int worldX = _floorMap[i][j].getPos()[0];
00163                 int worldY = _floorMap[i][j].getPos()[1];
00164 
00165                 // Checks if the player is close enough to the tile to render it to optimize memory
      and CPU usage
00166                 if (worldX + Const.WRLD_tileScreenSize > world.player.worldX - playerScreenX
00167                         && worldX - Const.WRLD_tileScreenSize < world.player.worldX + playerScreenX
00168                         && worldY + Const.WRLD_tileScreenSize > world.player.worldY - playerScreenY
00169                         && worldY - Const.WRLD_tileScreenSize < world.player.worldY + playerScreenY) {
00170                     _floorMap[i][j].updateFrames();
00171                     _topMap[i][j].updateFrames();
00172                 }
00173             }
00174         }
00175     }
```

Here is the call graph for this function: Here is the caller graph for this function:

8.38.4 Member Data Documentation

8.38.4.1 _floorMap

```
Tile [][] tiles.TileManager._floorMap  [private]
```

Definition at line 26 of file TileManager.java.

8.38.4.2 _floorMapTextures

```
Tile [] tiles.TileManager._floorMapTextures  [private]
```

Definition at line 23 of file TileManager.java.

8.38.4.3 _topMap

```
Tile [][] tiles.TileManager._topMap  [private]
```

Definition at line 27 of file TileManager.java.

8.38.4.4 _topMapTextures

```
Tile [] tiles.TileManager._topMapTextures  [private]
```

Definition at line 24 of file TileManager.java.

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

- src/tiles/TileManager.java

8.39 item.weapon.Weapon Class Reference

Inheritance diagram for item.weapon.Weapon:

Collaboration diagram for item.weapon.Weapon:

8.39.1 Detailed Description

Definition at line 5 of file Weapon.java.

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

- src/item/weapon/Weapon.java

8.40 CreationCombatScene.game.Window Class Reference

Inheritance diagram for CreationCombatScene.game.Window:

Collaboration diagram for CreationCombatScene.game.Window:

Public Member Functions

- Window ()
- void run ()
- void startGameThread ()
- void update ()
- void paintComponent (Graphics g)

Public Attributes

- Scene scene
- final int screenWidth =800
- final int screenHeight =600

Package Attributes

- final int FPS = 120
- Thread gameThread

8.40.1 Detailed Description

Definition at line 13 of file Window.java.

8.40.2 Constructor & Destructor Documentation

8.40.2.1 Window()

```
CreationCombatScene.game.Window.Window ( )
```

Definition at line 28 of file Window.java.

```
00028                     {
00029         PlayerTest playerTest=new PlayerTest();
00030         Enemy ennemy=new Enemy();
00031 
00032         scene = scene.fightScene(playerTest,ennemy);
00033         this.setPreferredSize(new Dimension(screenWidth,screenHeight));
00034         this.setBackground(Color.black);
00035         this.setDoubleBuffered(true);
00036         this.addKeyListener(scene.keyH);
00037         this.setFocusable(true);
00038 
00039 
00040     }
```

8.40.3 Member Function Documentation

8.40.3.1 paintComponent()

```
void CreationCombatScene.game.Window.paintComponent (
            Graphics g )
```

Definition at line 72 of file Window.java.

```
00072                                           {
00073 
00074         super.paintComponent(g);
00075 
00076         Graphics2D g2= (Graphics2D)g;
00077 
00078         scene.draw(g2,screenWidth,screenHeight);
00079 
00080         g2.dispose();
00081     }
```

Here is the call graph for this function:

8.40.3.2 run()

```
void CreationCombatScene.game.Window.run ( )
```

Definition at line 43 of file Window.java.

```
00043                     {
00044         double drawInterval = 10e9/FPS;
00045         long lastTime = System.nanoTime();
00046         long currentTime;
00047 
00048         while(gameThread != null){
00049             currentTime = System.nanoTime();
00050 
00051             scene.dt += (currentTime - lastTime) / drawInterval;
00052             lastTime = currentTime;
00053 
00054             if(scene.dt > 0.1){
00055                 update();           //update() method for window both updates the world and repaints
      the components of it
00056                 scene.dt-= 0.1;
00057             }
00058         }
00059     }
```

Here is the call graph for this function:

8.40.3.3 startGameThread()

```
void CreationCombatScene.game.Window.startGameThread ( )
```

Definition at line 61 of file Window.java.

```
00061                                {
00062         gameThread = new Thread(this);
00063         gameThread.start();
00064     }
```

Here is the caller graph for this function:

8.40.3.4 update()

```
void CreationCombatScene.game.Window.update ( )
```

Definition at line 66 of file Window.java.

```
00066                         {
00067         scene.update();         //Updating abstract class scene which means either world or fightscene
00068         repaint();
00069     }
```

Here is the call graph for this function: Here is the caller graph for this function:

8.40.4 Member Data Documentation

8.40.4.1 FPS

```
final int CreationCombatScene.game.Window.FPS = 120  [package]
```

Definition at line 23 of file Window.java.

8.40.4.2 gameThread

```
Thread CreationCombatScene.game.Window.gameThread  [package]
```

Definition at line 24 of file Window.java.

8.40.4.3 scene

```
Scene CreationCombatScene.game.Window.scene
```

Definition at line 15 of file Window.java.

8.40.4.4 screenHeight

```
final int CreationCombatScene.game.Window.screenHeight =600
```

Definition at line 19 of file Window.java.

8.40.4.5 screenWidth

```
final int CreationCombatScene.game.Window.screenWidth =800
```

Definition at line 18 of file Window.java.

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

- test/CreationCombatScene/game/Window.java

8.41 game.Window Class Reference

Represents the window that displays the game.

Inheritance diagram for game.Window:

Collaboration diagram for game.Window:

Public Member Functions

- Window ()
  Constructor for the Window class.
- void run ()
  Runs the game loop.
- void startGameThread ()
  Starts the game thread.
- void update ()
  Updates the game scene.
- void paintComponent (Graphics g)
  Paints the components of the game window.

Public Attributes

- Scene scene
  The game scene.
- final int screenWidth = 800
  The width of the game screen.
- final int screenHeight = 600
  The height of the game screen.

Package Attributes

- World world = World.getWorld()

  The game world.
- Thread gameThread

  The game thread.

Private Attributes

- final int _FPS = 60

  The frames per second for the game.

8.41.1 Detailed Description

Represents the window that displays the game.

Definition at line 20 of file Window.java.

8.41.2 Constructor & Destructor Documentation

8.41.2.1 Window()

```
game.Window.Window ( )
```

Constructor for the Window class.

Definition at line 54 of file Window.java.

```
00054                     {
00055         scene = World.getWorld();
00056         this.setPreferredSize(new Dimension(screenWidth, screenHeight));
00057         this.setBackground(Color.black);
00058         this.setDoubleBuffered(true);
00059         this.addKeyListener(scene.keyH);
00060         this.setFocusable(true);
00061         world.setupGame(); // Creation of instance setter
00062     }
```

Here is the call graph for this function:

8.41.3 Member Function Documentation

8.41.3.1 paintComponent()

```
void game.Window.paintComponent (
            Graphics g )
```

Paints the components of the game window.

Parameters

g	The Graphics object for drawing.

Definition at line 107 of file Window.java.

```
00107                                           {
00108         super.paintComponent(g);
00109
00110         Graphics2D g2 = (Graphics2D) g;
00111
00112         scene.draw(g2, screenWidth, screenHeight);
00113
00114         // Darkens the scene on the background to let the menu on top
00115         if (scene.state == State.PAUSE) {
00116             g2.setColor(new Color(0, 0, 0, 180));
00117             g2.fillRect(0, 0, screenWidth, screenHeight);
00118             scene.menu.draw(g2, screenWidth, screenHeight);
00119         }
00120
00121         g2.dispose();
00122     }
```

Here is the call graph for this function:

8.41.3.2 run()

```
void game.Window.run ( )
```

Runs the game loop.

Definition at line 68 of file Window.java.

```
00068                           {
00069         double drawInterval = 10e9 / _FPS;
00070         long lastTime = System.nanoTime();
00071         long currentTime;
00072
00073         while (gameThread != null) {
00074             currentTime = System.nanoTime();
00075
00076             Scene.dt += (currentTime - lastTime) / drawInterval;
00077             lastTime = currentTime;
00078
00079             if (Scene.dt > 0.1) {
00080                 update(); // update() method for window both updates the world and repaints the
      components of it
00081                 Scene.dt -= 0.05;
00082             }
00083         }
00084     }
```

Here is the call graph for this function:

8.41.3.3 startGameThread()

```
void game.Window.startGameThread ( )
```

Starts the game thread.

Definition at line 89 of file Window.java.

```
00089                                   {
00090         gameThread = new Thread(this);
00091         gameThread.start();
00092     }
```

Here is the caller graph for this function:

8.41.3.4 update()

```
void game.Window.update ( )
```

Updates the game scene.

Definition at line 97 of file Window.java.

```
00097                     {
00098         scene.update(); // Updates the whole world's props & animations
00099         repaint();
00100     }
```

Here is the call graph for this function: Here is the caller graph for this function:

8.41.4 Member Data Documentation

8.41.4.1 _FPS

```
final int game.Window._FPS = 60  [private]
```

The frames per second for the game.

Definition at line 44 of file Window.java.

8.41.4.2 gameThread

```
Thread game.Window.gameThread  [package]
```

The game thread.

Definition at line 49 of file Window.java.

8.41.4.3 scene

```
Scene game.Window.scene
```

The game scene.

Definition at line 24 of file Window.java.

8.41.4.4 screenHeight

```
final int game.Window.screenHeight = 600
```

The height of the game screen.

Definition at line 34 of file Window.java.

8.41.4.5 screenWidth

```
final int game.Window.screenWidth = 800
```

The width of the game screen.

Definition at line 29 of file Window.java.

8.41.4.6 world

```
World game.Window.world = World.getWorld()  [package]
```

The game world.

Definition at line 39 of file Window.java.

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

- src/game/Window.java

8.42 game.World Class Reference

Represents the game world and manages its entities.

Inheritance diagram for game.World:

Collaboration diagram for game.World:

Public Member Functions

- void setupGame ()
  Set up the initial state of the game.
- void addObject (Point coordinates, Props object)
  Adds an object to the HashMap with the specified coordinates.
- void addEnemy (Point coordinates, Enemy enemy)
- void update ()
  Updates the game world based on the scene state.
- void draw (Graphics2D g2, int screenWidth, int screenHeight)
  Draws the game world and its entities.

Static Public Member Functions

- static World getWorld ()

  Gets the instance of the World.

Public Attributes

- TileManager tileManager = new TileManager(this)
- EntitySetter entitySetter = new EntitySetter(this)
- HashMap< Point, Props > objMap = new HashMap<>()
- Player player

Static Public Attributes

- static HashMap< Point, Enemy > enemies = new HashMap<>()

Private Attributes

- FightScene _currfight

Static Private Attributes

- static World _instance

Additional Inherited Members

8.42.1 Detailed Description

Represents the game world and manages its entities.

Definition at line 24 of file World.java.

8.42.2 Member Function Documentation

8.42.2.1 addEnemy()

```
void game.World.addEnemy (
            Point coordinates,
            Enemy enemy )
```

Definition at line 84 of file World.java.

```
00084                                                   {
00085         enemies.put(coordinates,enemy);
00086     }
```

Here is the caller graph for this function:

8.42.2.2 addObject()

```
void game.World.addObject (
            Point coordinates,
            Props object )
```

Adds an object to the HashMap with the specified coordinates.

Parameters

coordinates	The coordinates of the object.
object	The object to be added.

Definition at line 80 of file World.java.

```
00080                                                                   {
00081         objMap.put(coordinates, object);
00082     }
```

Here is the caller graph for this function:

8.42.2.3 draw()

```
void game.World.draw (
             Graphics2D g2,
             int screenWidth,
             int screenHeight )
```

Draws the game world and its entities.

Parameters

g2	The Graphics2D object for drawing.
screenWidth	The width of the screen.
screenHeight	The height of the screen.

Reimplemented from game.Scene.

Definition at line 134 of file World.java.

```
00134                                                                                   {
00135
00136         //If there is a fight, draw the fight instead of the game world
00137         if(_currfight != null){
00138             if( _currfight.state == FightState.FIGHTING){
00139                 _currfight.draw(g2);
00140             }
00141             if(_currfight.state == FightState.WON){
00142                 _currfight = null;
00143             }
00144         }
00145
00146         else{
00147             // TILE
00148             tileManager.draw(g2, this);
00149             // OBJECT
00150             for (Props props :  objMap.values()) {
00151                 props.draw(g2, this);
00152             }
00153             // PLAYER
00154             player.drawInWorld(g2, (screenWidth-Const.WRLD_entityScreenSize)/2,
00155                                    (screenHeight-Const.WRLD_entityScreenSize)/2); // Player is always
      centered to screen
00156
00157
00158
00159             //ENEMIES
00160             for(Enemy enemy :  enemies.values()){
00161                 // Calculate the screen position of the character relative to the player's position
00162                 int screenX = enemy.worldX - player.worldX + (Const.WDW_width -
      Const.WRLD_entityScreenSize)/2;
00163                 int screenY = enemy.worldY - player.worldY + (Const.WDW_height -
      Const.WRLD_entityScreenSize)/2;
00164
00165
00166
```

```
00167                 enemy.drawInWorld(g2, screenX, screenY);
00168             }
00169         }
00170     }
```

Here is the call graph for this function:

8.42.2.4 getWorld()

```
static World game.World.getWorld ( )  [static]
```

Gets the instance of the World.

Returns

The World instance.

Definition at line 50 of file World.java.

```
00050                                   {
00051         Textbox.loadFont("rainyhearts");
00052         if (_instance == null) {
00053             _instance = new World();
00054             _instance.state = State.WORLD;
00055         }
00056         return _instance;
00057     }
```

Here is the call graph for this function: Here is the caller graph for this function:

8.42.2.5 setupGame()

```
void game.World.setupGame ( )
```

Set up the initial state of the game.

Definition at line 63 of file World.java.

```
00063                                 {
00064         entitySetter.setObject();
00065         entitySetter.setEnemies();
00066
00067         player = new Player("player", 15 * Const.WRLD_tileScreenSize,
00068             15 * Const.WRLD_tileScreenSize, 0, 0, 0, "down", 4, 20);
00069
00070     }
```

Here is the call graph for this function: Here is the caller graph for this function:

8.42.2.6 update()

```
void game.World.update ( )
```

Updates the game world based on the scene state.

Reimplemented from game.Scene.

Definition at line 91 of file World.java.

```
00091                         {
00092         checkPauseScene();
00093         if (state == State.WORLD) {
00094             Point ptn = new Point((int)13 * Const.WRLD_entityScreenSize, (int) 13 *
      Const.WRLD_entityScreenSize);
00095             Props key = objMap.get(ptn);
00096             //System.out.println("("+key.worldX +"," + key.worldY + ")  (" + key.hitbox.x
      +","+key.hitbox.y+")");
00097             //int playerScreenX = (Const.WDW_width - Const.WRLD_entityScreenSize) / 2;
00098             //int playerScreenY = (Const.WDW_height - Const.WRLD_entityScreenSize) / 2;
00099 
00100             player.update(this, dt);
00101             tileManager.update(this);
00102 
00103 
00104             //Update enemy if he's close enough otherwise useless to update (5 tile radius around the
      screen)
00105             for(Enemy enemy :  enemies.values()){
00106                 int playerScreenX = (Const.WDW_width - Const.WRLD_entityScreenSize) / 2;
00107                 int playerScreenY = (Const.WDW_height - Const.WRLD_entityScreenSize) / 2;
00108 
00109                 if (enemy.worldX + 5*Const.WRLD_tileScreenSize > player.worldX - playerScreenX
00110                 && enemy.worldX - 5*Const.WRLD_tileScreenSize < player.worldX + playerScreenX
00111                 && enemy.worldY + 5*Const.WRLD_tileScreenSize > player.worldY - playerScreenY
00112                 && enemy.worldY - 5*Const.WRLD_tileScreenSize < player.worldY + playerScreenY) {
00113                     enemy.update(this, dt);
00114 
00115                 }
00116             }
00117         }
00118 
00119 
00120 
00121 
00122         if (state == State.FIGHT){
00123             _currfight.update(this);
00124         }
00125     }
```

Here is the call graph for this function:

8.42.3 Member Data Documentation

8.42.3.1 _currfight

```
FightScene game.World._currfight  [private]
```

Definition at line 27 of file World.java.

8.42.3.2 _instance

```
World game.World._instance  [static], [private]
```

Definition at line 26 of file World.java.

8.42.3.3 enemies

```
HashMap<Point, Enemy> game.World.enemies = new HashMap<>()  [static]
```

Definition at line 36 of file World.java.

8.42.3.4 entitySetter

```
EntitySetter game.World.entitySetter = new EntitySetter(this)
```

Definition at line 32 of file World.java.

8.42.3.5 objMap

```
HashMap<Point, Props> game.World.objMap = new HashMap<>()
```

Definition at line 35 of file World.java.

8.42.3.6 player

```
Player game.World.player
```

Initial value:

```
= new Player("player",15 * Const.WRLD_tileScreenSize,
            15 * Const.WRLD_tileScreenSize, 0, 0, 0, "down", 4, 20)
```

Definition at line 40 of file World.java.

8.42.3.7 tileManager

```
TileManager game.World.tileManager = new TileManager(this)
```

Definition at line 28 of file World.java.

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

- src/game/World.java

Chapter 9

File Documentation

9.1 aFaire.md File Reference

9.2 README.md File Reference

9.3 src/entity/Character.java File Reference

This file contains the implementation of the Character class, representing an abstract character entity with position, hitbox, stats, and animations.

Classes

- class entity.Character

  Represents an abstract character entity with position, hitbox, stats, and animations.

Packages

- package entity

9.3.1 Detailed Description

This file contains the implementation of the Character class, representing an abstract character entity with position, hitbox, stats, and animations.

Definition in file Character.java.

9.4 Character.java

Go to the documentation of this file.

```
00001
00006 package entity;
00007
00008 import game.Const;
00009 import game.Scene;
00010 import game.World;
00011 import tiles.TileManager;
00012 import game.Scene.State;
00013 import java.awt.*;
00014 import java.awt.image.BufferedImage;
00015 import java.io.FileInputStream;
00016 import java.io.IOException;
00017
00018 import javax.imageio.ImageIO;
00019
00025 public abstract class Character extends Entity {
00026     int health;
00027     int mana;
00028     int agility;
00029     int strength;
00030     int defense;
00031     int initiative;
00032     public int speed;
00033     public int dirX, dirY;
00034     int hasKey = 0; //normalement dans player
00035
00036     // Which direction is the entity facing (if directions are available) for animation
00037     public String facing;
00038
00039     // Character animations
00040     private BufferedImage[] _idle_up;
00041     private BufferedImage[] _idle_down;
00042     private BufferedImage[] _idle_right;
00043     private BufferedImage[] _idle_left;
00044     private BufferedImage[] _walk_up;
00045     private BufferedImage[] _walk_down;
00046     private BufferedImage[] _walk_right;
00047     private BufferedImage[] _walk_left;
00048
00060     public Character(String entityName, int x, int y, int dirX, int dirY, int speed, String facing,
      int _spriteCntMax, int spriteSpeed) {
00061         super(entityName, x, y, _spriteCntMax, spriteSpeed,true);
00062
00063
00064
00065         // Hitbox settings (size of the entity)
00066         this.hitbox.width = Const.WRLD_entityScreenSize / 2;
00067         this.hitbox.height = Const.WRLD_entityScreenSize / 2;
00068         this.dirX = dirX;
00069         this.dirY = dirY;
00070         this.speed = speed;
00071         this.facing = facing;
00072
00073         _idle_up = new BufferedImage[_spriteCntMax];
00074         _idle_down = new BufferedImage[_spriteCntMax];
00075         _idle_right = new BufferedImage[_spriteCntMax];
00076         _idle_left = new BufferedImage[_spriteCntMax];
00077         _walk_up = new BufferedImage[_spriteCntMax];
00078         _walk_down = new BufferedImage[_spriteCntMax];
00079         _walk_right = new BufferedImage[_spriteCntMax];
00080         _walk_left = new BufferedImage[_spriteCntMax];
00081         loadTextures(entityName);
00082     }
00083
00089     @Override
00090     public void update(Scene scene, double dt) {
00091         if (scene.state == State.WORLD) {
00092             World currWorld = World.getWorld();
00093
00094             hitbox.x = worldX + hitbox.width / 2;
00095             hitbox.y = worldY + hitbox.height;
00096
00097             // CHECK THE COLLISION
00098             move(World.getWorld(), speed, dt);
00099             checkTileCollision(currWorld.tileManager);
00100
00101
00102
00103             updateFrames();
00104         }
00105     }
00106
```

```
00111     private void checkTileCollision(TileManager tileManager) {
00112         // Checking tiles with hitbox
00113 
00114 
00115         if((tileManager.getTile(hitbox.x, hitbox.y - 5).getCollision() //Checks collision with tile on
      top of the character
00116         || tileManager.getTile(hitbox.x + hitbox.width , hitbox.y - 5).getCollision() )&&  dirY == -1)
      {
00117             worldY = tileManager.getTile(hitbox.x,hitbox.y).getPos()[1] - hitbox.height;    //Prevent
      moving if collidable terrain
00118         }
00119         if((tileManager.getTile(hitbox.x, hitbox.y + hitbox.height + 5).getCollision() //Checks
      collision with tile beneath of the character
00120         || tileManager.getTile(hitbox.x + hitbox.width, hitbox.y + hitbox.height + 5).getCollision())
      && dirY == 1){
00121             worldY = tileManager.getTile(hitbox.x,hitbox.y).getPos()[1] -1;        //Prevent moving if
      collidable terrain
00122         }
00123         if((tileManager.getTile(hitbox.x - 5, hitbox.y).getCollision() //Checks collision with tile on
      the left of the character
00124         || tileManager.getTile(hitbox.x - 5, hitbox.y + hitbox.height).getCollision()) && dirX == -1)
      {
00125             worldX = tileManager.getTile(hitbox.x,hitbox.y).getPos()[0] - hitbox.width/2 ;
      //Prevent moving if collidable terrain
00126         }
00127         if((tileManager.getTile(hitbox.x + hitbox.width + 5, hitbox.y).getCollision() //Checks
      collision with tile on the right of the character
00128         || tileManager.getTile(hitbox.x + hitbox.width + 5, hitbox.y + hitbox.height).getCollision())
      && dirX == 1){
00129             worldX = tileManager.getTile(hitbox.x,hitbox.y).getPos()[0] + hitbox.width/2 - 1;
      //Prevent moving if collidable terrain
00130         }
00131     }
00132 
00133 
00134 
00141     protected void move(World world, int speed, double dt) {
00142         if ((dirX == 0 && dirY != 0) || (dirY == 0 && dirX != 0)) {
00143             if (dirX == 1) {
00144                 worldX += speed * dt;
00145             }
00146             if (dirX == -1) {
00147                 worldX -= speed * dt;
00148             }
00149             if (dirY == 1) {
00150                 worldY += speed * dt;
00151             }
00152             if (dirY == -1) {
00153                 worldY -= speed * dt;
00154             }
00155         }
00156         if (dirX != 0 && dirY != 0) {
00157             double normSum = Math.sqrt(dirX * dirX + dirY * dirY);      //Normalizing vector
00158             worldX += (dirX / normSum) * speed * dt;
00159             worldY += (dirY / normSum) * speed * dt;
00160         }
00161     }
00162 
00168     protected void accelerate(int maxSpeed, int factor, double dt) {
00169         if (speed < maxSpeed) {
00170             speed += factor*dt;
00171         }
00172         if (speed > maxSpeed) {
00173             speed = maxSpeed;
00174         }
00175     }
00176 
00181     protected void decelerate(int factor,double dt) {
00182         speed -= factor*dt;
00183     }
00184 
00189     protected void loadTextures(String name) {
00190         try {
00191             for (int i = 0; i < _spriteCntMax; i++) {
00192                 _idle_up[i] = ImageIO.read(new FileInputStream("res/entity/character/idle/" + name +
      "/up" + (i + 1) + ".png"));
00193                 _idle_down[i] = ImageIO.read(new FileInputStream("res/entity/character/idle/" + name +
      "/down" + (i + 1) + ".png"));
00194                 _idle_left[i] = ImageIO.read(new FileInputStream("res/entity/character/idle/" + name +
      "/left" + (i + 1) + ".png"));
00195                 _idle_right[i] = ImageIO.read(new FileInputStream("res/entity/character/idle/" + name
      + "/right" + (i + 1) + ".png"));
00196 
00197                 _walk_up[i] = ImageIO.read(new FileInputStream("res/entity/character/walk/" + name +
      "/up" + (i + 1) + ".png"));
00198                 _walk_down[i] = ImageIO.read(new FileInputStream("res/entity/character/walk/" + name +
      "/down" + (i + 1) + ".png"));
```

```
00199                 _walk_left[i] = ImageIO.read(new FileInputStream("res/entity/character/walk/" + name +
      "/left" + (i + 1) + ".png"));
00200                 _walk_right[i] = ImageIO.read(new FileInputStream("res/entity/character/walk/" + name
      + "/right" + (i + 1) + ".png"));
00201             }
00202
00203         } catch (IOException e) {
00204             e.printStackTrace();
00205         }
00206     }
00207
00208
00215     public void drawInWorld(Graphics2D g2, int screenX, int screenY) {
00216         BufferedImage image = null;
00217         if (speed == 0) { // IDLE ANIMATIONS
00218             for (int i = 0; i < _spriteCntMax; i++) {
00219                 switch (facing) {
00220                     case "up":
00221                         if (_spriteCnt == i) image = _idle_up[i];
00222                         break;
00223                     case "down":
00224                         if (_spriteCnt == i) image = _idle_down[i];
00225                         break;
00226                     case "left":
00227                         if (_spriteCnt == i) image = _idle_left[i];
00228                         break;
00229                     case "right":
00230                         if (_spriteCnt == i) image = _idle_right[i];
00231                         break;
00232                 }
00233             }
00234         }
00235         if (speed > 0) { // WALKING ANIMATIONS
00236             for (int i = 0; i < _spriteCntMax; i++) {
00237                 switch (facing) {
00238                     case "up":
00239                         if (_spriteCnt == i) image = _walk_up[i];
00240                         break;
00241                     case "down":
00242                         if (_spriteCnt == i) image = _walk_down[i];
00243                         break;
00244                     case "left":
00245                         if (_spriteCnt == i) image = _walk_left[i];
00246                         break;
00247                     case "right":
00248                         if (_spriteCnt == i) image = _walk_right[i];
00249                         break;
00250                 }
00251             }
00252         }
00253
00254
00255         int playerScreenX = (Const.WDW_width - Const.WRLD_entityScreenSize) / 2;
00256         int playerScreenY = (Const.WDW_height - Const.WRLD_entityScreenSize) / 2;
00257
00258         //Checking if we need to draw enemy or not
00259         if (worldX + Const.WRLD_tileScreenSize > worldX - playerScreenX
00260         && worldX - Const.WRLD_tileScreenSize < worldX + playerScreenX
00261         && worldY + Const.WRLD_tileScreenSize > worldY - playerScreenY
00262         && worldY - Const.WRLD_tileScreenSize < worldY + playerScreenY) {
00263
00264             g2.drawImage(image, screenX, screenY, Const.WRLD_entityScreenSize,
      Const.WRLD_entityScreenSize, null);
00265             g2.drawRect(screenX + hitbox.width / 2, screenY + hitbox.height, hitbox.width,
      hitbox.height); // Center the hitbox to the entity
00266         }
00267     }
00268
00269
00276     public void drawInFight(Graphics2D g2, int screenX, int screenY) {
00277         // Other function to draw in fight scene
00278     }
00279
00280     @Override
00281     protected void playerInterraction(Player player) {
00282     }
00283 }
```

9.5 src/entity/Enemy.java File Reference

This file contains the implementation of the Enemy class, representing an enemy entity extending the Character class.

Classes

- class entity.Enemy

  Represents an enemy entity in the game.

Packages

- package entity

9.5.1 Detailed Description

This file contains the implementation of the Enemy class, representing an enemy entity extending the Character class.

Definition in file Enemy.java.

9.6 Enemy.java

Go to the documentation of this file.

```
00001
00006 package entity;
00007
00008 import game.Scene;
00009
00015 public class Enemy extends Character {
00016
00017     private int _xpRate; // Experience points rate for defeating the enemy
00018     public String name;
00030     public Enemy(String enemyName, int worldX, int worldY, int dirX, int dirY, int speed, String
      facing, int spriteCntMax, int spriteSpeed) {
00031         super(enemyName, worldX, worldY, dirX, dirY, speed, facing, spriteCntMax, spriteSpeed);  //
      Calls the parent class for entity setup, specifying scene.keyH for player
00032     }
00033
00038     public boolean touchingPlayer(Player player){
00039
00040         if((hitbox.x >= player.hitbox.x + player.hitbox.width)      // trop à droite
00041         || (hitbox.x + hitbox.width <= player.hitbox.x) // trop à gauche
00042         || (hitbox.y >= player.hitbox.y + player.hitbox.height) // trop en bas
00043         || (hitbox.y + hitbox.height <= player.hitbox.y)){// trop en haut
00044             return false;
00045         }
00046
00047         return true;
00048     }
00049
00050
00056     @Override
00057     public void update(Scene scene, double dt) {
00058         super.update(scene, dt); // Calls the parent class update method
00059         //TODO : find a method to make the enemy move in predictive patterns
00060     }
00061
00062     public void playerInterraction(Player player){
00063
00064     }
00065 }
```

9.7 src/entity/Entity.java File Reference

This file contains the implementation of the Entity class, representing an abstract entity with position, hitbox, and animations.

Classes

- class entity.Entity

  Represents an abstract entity with position, hitbox, and animations.

Packages

- package entity

9.7.1 Detailed Description

This file contains the implementation of the Entity class, representing an abstract entity with position, hitbox, and animations.

Definition in file Entity.java.

9.8 Entity.java

Go to the documentation of this file.

```
00001
00006 package entity;
00007
00008 import java.awt.Rectangle;
00009 import game.Scene;
00010
00015 public abstract class Entity {
00016     // Position of the entity in the world, directions, and speed
00017     public int worldX, worldY;
00018     protected boolean collision;
00019     // Hitbox of the entity
00020     public Rectangle hitbox = new Rectangle();
00021
00022     // Entity animations
00023     protected int _spriteCnt = 0; // Variable responsible for the incrementation of the different
    sprites
00024     protected int _spriteUpdater = 0; // Variable responsible for the incrementation of the speed of
    the sprites
00025     protected int _spriteSpeed; // How fast the sprites are changing (higher spriteSpeed means slower
    time to change)
00026     protected int _spriteCntMax; // How many sprites does the entity have
00027
00028     public String name; //Name of the current entity
00029
00033     public Entity() {
00034         // Default constructor
00035     }
00036
00044     public Entity(String entityName, int x, int y, int _spriteCntMax, int spriteSpeed, boolean
    collision) {
00045         this.collision = collision;
00046         this.worldX = x;
00047         this.worldY = y;
00048         this.hitbox.x = worldX;
00049         this.hitbox.y = worldY;
00050         this._spriteCntMax = _spriteCntMax;
00051         this._spriteSpeed = spriteSpeed;
00052         this.name = entityName;
00053     }
00054
00060     public void update(Scene scene, double dt) {
00061         // Updating entity position accurately (at any point in time either pressing keys or not)
00062     }
00063
00064     // GRAPHICS
00065
00070     private void loadTextures() {
00071         // TODO: Different texture loading from characters
00072     }
00073
```

```
00077     protected void updateFrames() {
00078         _spriteUpdater++;
00079         if (_spriteUpdater > _spriteSpeed) {
00080             _spriteCnt++;
00081             if (_spriteCnt == _spriteCntMax) {
00082                 _spriteCnt = 0;
00083             }
00084             _spriteUpdater = 0;
00085         }
00086     }
00087
00088     protected abstract void playerInterraction(Player player);
00089 }
```

9.9 src/entity/EntitySetter.java File Reference

This file contains the implementation of the EntitySetter class, responsible for initializing and setting up objects (entities) in the game world.

Classes

- class entity.EntitySetter

  Responsible for initializing and setting up objects (entities) in the game world.

Packages

- package entity

9.9.1 Detailed Description

This file contains the implementation of the EntitySetter class, responsible for initializing and setting up objects (entities) in the game world.

Definition in file EntitySetter.java.

9.10 EntitySetter.java

Go to the documentation of this file.

```
00001
00006 package entity;
00007
00008 import entity.props.*;
00009 import game.World;
00010 import game.Const;
00011 import java.awt.*;
00012
00017 public class EntitySetter {
00018     World world;
00019
00024     public EntitySetter(World world) {
00025         this.world = world;
00026     }
00027
00031     public void setObject() {
00032         // Create and set up a Key object at a specific location in the world
00033         Props key = new OBJ_Key(13 * Const.WRLD_entityScreenSize,13 * Const.WRLD_entityScreenSize);
00034         world.addObject(new Point((int) key.worldX, (int) key.worldY), key);
00035
00036         // Create and set up a Door object at a specific location in the world
```

```
00037         Props door = new OBJ_Door(14 * Const.WRLD_entityScreenSize,13 * Const.WRLD_entityScreenSize);
00038         world.addObject(new Point((int) door.worldX, (int) door.worldY), door);
00039 
00040         // Create and set up a Chest object at a specific location in the world
00041         Props chest = new OBJ_Chest(15 * Const.WRLD_entityScreenSize,13 *
      Const.WRLD_entityScreenSize);
00042         world.addObject(new Point((int) chest.worldX, (int) chest.worldY), chest);
00043     }
00044 
00045     public void setEnemies(){
00046         Enemy enemy1 = new Enemy("orc", 8*Const.WRLD_entityScreenSize, 10*Const.WRLD_entityScreenSize,
      0, 0, 0, "down", 4, 20);
00047         world.addEnemy(new Point((int) enemy1.worldX, (int) enemy1.worldY), enemy1);
00048 
00049         Enemy enemy2 = new Enemy("orc", 16*Const.WRLD_entityScreenSize,
      10*Const.WRLD_entityScreenSize, 0, 0, 0, "up", 4, 20);
00050         world.addEnemy(new Point((int) enemy2.worldX, (int) enemy2.worldY), enemy2);
00051 
00052     }
00053 }
```

9.11 src/entity/Player.java File Reference

This file contains the implementation of the Player class, representing a player entity extending the Character class.

Classes

- class entity.Player

  Represents the player entity in the game.

Packages

- package entity

9.11.1 Detailed Description

This file contains the implementation of the Player class, representing a player entity extending the Character class.

Definition in file Player.java.

9.12 Player.java

Go to the documentation of this file.

```
00001 
00006 package entity;
00007 
00008 import entity.props.Props;
00009 import game.Scene;
00010 import game.Scene.State;
00011 import game.World;
00012 import item.Item;
00013 
00014 import java.awt.*;
00015 
00021 public class Player extends Character {
00022 
00023     public int hasKey = 0;
00035     public Player(String entityName, int worldX, int worldY, int dirX, int dirY, int speed, String
      facing, int spriteCntMax, int spriteSpeed) {
```

```
00036         super(entityName, worldX, worldY, dirX, dirY, speed, facing, spriteCntMax, spriteSpeed);  //
      Calls the parent class for entity setup, specifying scene.keyH for player
00037     }
00038
00044     public void update(Scene scene, double dt) {
00045         super.update(scene, dt);  // Calls the parent class update method
00046         // World updates
00047         if (scene.state == State.WORLD) {
00048             if (scene.keyH.upPressed || scene.keyH.downPressed || scene.keyH.leftPressed ||
      scene.keyH.rightPressed) {
00049                 dirX = 0;
00050                 dirY = 0;
00051                 if (scene.keyH.leftPressed) {
00052                     dirX = -1;
00053                     facing = "left";
00054                 }
00055                 if (scene.keyH.rightPressed) {
00056                     dirX = 1;
00057                     facing = "right";
00058                 }
00059                 if (scene.keyH.upPressed) {
00060                     dirY = -1;
00061                     facing = "up";
00062                 }
00063                 if (scene.keyH.downPressed) {
00064                     dirY = 1;
00065                     facing = "down";
00066                 }
00067                 accelerate(30,20, dt);
00068             } else {
00069                 if (speed > 0) {
00070                     decelerate(1,dt);
00071                 }
00072             }
00073
00074             // CHECK OBJECT COLLISION
00075             Point objIndex = checkObject(this, World.getWorld());
00076             pickUpObject(World.getWorld(), objIndex);
00077
00078
00079         }
00080
00081         // Fightscene updates
00082         if (scene.state == State.FIGHT) {
00083             // To be implemented
00084
00085
00086         }
00087     }
00088
00095     public Point checkObject(Entity entity, World world) {
00096
00097         Point index = null;
00098
00099         for (Props obj :  world.objMap.values()) {
00100             if (obj != null) {
00101                 if (entity.hitbox.intersects(obj.hitbox)) {
00102                     if (obj.getCollision()) {   //If object has "solid" collision
00103                         //prevent the player from moving in the hitbox
00104
00105                     }
00106                     index = new Point((int) obj.worldX, (int) obj.worldY);
00107                     break;
00108                 }
00109             }
00110         }
00111
00112         return index;
00113     }
00114
00115
00116
00117     public void pickUpObject(World gp, Point index) {
00118         if (index != null) {
00119             Entity object = gp.objMap.get(index);
00120             object.playerInterraction(this);
00121         }
00122     }
00123
00124     public void addItem(Item i){
00125         System.out.println("J'ai rajoutée un item");
00126     }
00127
00128 }
```

9.13 src/entity/props/OBJ_Chest.java File Reference

This file contains the implementation of the OBJ_Chest class, which represents a chest object extending the Props class.

Classes

- class entity.props.OBJ_Chest

  Represents a chest object.

Packages

- package entity.props

9.13.1 Detailed Description

This file contains the implementation of the OBJ_Chest class, which represents a chest object extending the Props class.

Definition in file OBJ_Chest.java.

9.14 OBJ_Chest.java

Go to the documentation of this file.

```
00001
00006 package entity.props;
00007
00008
00009 import entity.Player;
00010 import item.Generator;
00011 import item.Item;
00012
00018 public class OBJ_Chest extends Props {
00019
00024     public OBJ_Chest(int worldX, int worldY){
00025         super(worldX,worldY,"chest",1,0,true);
00026         loadTextures("chest");
00027         collision = true;
00028     }
00029     public void playerInterraction(Player p) {
00030         Item item= Generator.generateItem();
00031         if(true){
00032             p.addItem(item);
00033         }
00034         System.out.println("Chest interaction");
00035
00036         destroySelf();
00037
00038     }
00039 }
```

9.15 src/entity/props/OBJ_Door.java File Reference

This file contains the implementation of the OBJ_Door class, representing a door object extending the Props class.

Classes

- class entity.props.OBJ_Door

  Represents a door object in the game.

Packages

- package entity.props

9.15.1 Detailed Description

This file contains the implementation of the OBJ_Door class, representing a door object extending the Props class.

Definition in file OBJ_Door.java.

9.16 OBJ_Door.java

Go to the documentation of this file.

```
00001
00006 package entity.props;
00007
00008
00009 import entity.Player;
00010
00016 public class OBJ_Door extends Props {
00017
00022     public OBJ_Door(int worldX, int worldY) {
00023         super(worldX, worldY,"door",1,0, true);
00024
00025         loadTextures("door");
00026         collision = true;
00027     }
00028     public void playerInterraction(Player p) {
00029         if(p.hasKey > 0 ) {
00030             // TODO : Change door collision; it's still here but with different textures and
    properties
00031             p.hasKey--;
00032
00033             destroySelf();
00034         }
00035
00036         System.out.println("Key:"+p.hasKey);
00037
00038
00039
00040
00041     }
00042
00043 }
```

9.17 src/entity/props/OBJ_Key.java File Reference

This file contains the implementation of the OBJ_Key class, representing a key object extending the Props class.

Classes

- class entity.props.OBJ_Key

  Represents a key object in the game.

Packages

- package entity.props

9.17.1 Detailed Description

This file contains the implementation of the OBJ_Key class, representing a key object extending the Props class.

Definition in file OBJ_Key.java.

9.18 OBJ_Key.java

Go to the documentation of this file.

```
00001
00006 package entity.props;
00007
00008
00009 import entity.Player;
00010
00016 public class OBJ_Key extends Props {
00017
00022     public OBJ_Key(int worldX, int worldY) {
00023         super(worldX, worldY,"key",1,0,false);
00024
00025         loadTextures("key");
00026     }
00027     public void playerInterraction(Player p) {
00028
00029         p.hasKey++;
00030         System.out.println("Key:"+p.hasKey);
00031
00032         destroySelf();
00033     }
00034 }
```

9.19 src/entity/props/Props.java File Reference

This file contains the implementation of the Props class, representing in-game props with properties such as image, name, and position.

Classes

- class entity.props.Props

  Represents in-game props with properties such as image, name, and position.

Packages

- package entity.props

9.19.1 Detailed Description

This file contains the implementation of the Props class, representing in-game props with properties such as image, name, and position.

Definition in file Props.java.

9.20 Props.java

Go to the documentation of this file.

```
00001
00006 package entity.props;
00007
00008 import entity.Entity;
00009 import game.Const;
00010 import game.World;
00011
00012 import java.awt.*;
00013 import java.awt.image.BufferedImage;
00014 import java.io.FileInputStream;
00015 import java.io.IOException;
00016
00017 import javax.imageio.ImageIO;
00018
00024 public abstract class Props extends Entity {
00025     private BufferedImage image; // Image representing the prop
00026
00027
00028     Props(int x, int y, String name, int spriteCntMax, int spriteSpeed, boolean collision){
00029         super(name,x,y,spriteCntMax,spriteSpeed,collision);
00030         hitbox.width = 3*Const.WRLD_tileScreenSize/4;  //Slightly smaller than a tile
00031         hitbox.height = 3*Const.WRLD_tileScreenSize/4;
00032         hitbox.x = worldX + hitbox.width/4;
00033         hitbox.y = worldY + hitbox.height/4;
00034     }
00035
00036     public boolean getCollision(){
00037         return collision;
00038     }
00039
00040     protected void loadTextures(String name){
00041         try {
00042             // Load the image of the key from the specified file path
00043             image = ImageIO.read(new FileInputStream("res/object/"+name+".png"));
00044         } catch (IOException e) {
00045             // Print the stack trace in case of an IOException during image loading
00046             e.printStackTrace();
00047         }
00048     }
00049
00050
00051
00057     public void draw(Graphics2D g2, World world) {
00058         // Calculate the screen position of the player
00059         int playerScreenX = (Const.WDW_width - Const.WRLD_entityScreenSize) / 2;
00060         int playerScreenY = (Const.WDW_height - Const.WRLD_entityScreenSize) / 2;
00061
00062         // Calculate the screen position of the prop relative to the player's position
00063         int screenX = worldX - world.player.worldX + playerScreenX;
00064         int screenY = worldY - world.player.worldY + playerScreenY;
00065
00066         // Check if the prop is within the visible screen region around the player
00067         if (worldX + Const.WRLD_tileScreenSize > world.player.worldX - playerScreenX
00068                 && worldX - Const.WRLD_tileScreenSize < world.player.worldX + playerScreenX
00069                 && worldY + Const.WRLD_tileScreenSize > world.player.worldY - playerScreenY
00070                 && worldY - Const.WRLD_tileScreenSize < world.player.worldY + playerScreenY) {
00071             // Draw the prop on the screen
00072             g2.drawImage(image, screenX, screenY, Const.WRLD_entityScreenSize,
      Const.WRLD_entityScreenSize, null);
00073             g2.drawRect(screenX + Const.WRLD_entityScreenSize/8, screenY +
      Const.WRLD_entityScreenSize/8, this.hitbox.width, this.hitbox.height);
00074         }
00075     }
00076
00077     public void destroySelf(){
00078         Point point=new Point(worldX,worldY);
00079         World instance=World.getWorld();
00080         instance.objMap.remove(point,this);
00081     }
00082 }
```

9.21 src/game/Const.java File Reference

Classes

- class game.Const

Packages

- package game

9.22 Const.java

Go to the documentation of this file.

```
00001 package game;
00002
00003 public final class Const {
00004
00005     //Window parameters
00006     public final static int WDW_width = 800;
00007     public final static int WDW_height = 600;
00008
00009     //World parameters
00010     public final static int WRLD_tileSize = 16;
00011     public final static int WRLD_scale = 3;
00012     public final static int WRLD_tileScreenSize = WRLD_tileSize*WRLD_scale;
00013     public final static int WRLD_entityScreenSize = WRLD_tileSize*WRLD_scale;
00014     // World initialization settings
00015     public final static int WRLD_maxRow = 27, WRLD_maxCol = 27; // DONT FORGET TO MODIFY WHEN CHANGING
      THE MAP !!!
00016
00017     //FightScene parameters
00018     public final static int FGHT_entityScreenSize = 200;
00019
00020     //TileManager
00021     public final static int nbFloorTextures = 6;
00022     public final static int nbTopTextures = 29;
00023 }
```

9.23 src/game/FightScene.java File Reference

This file contains the implementation of the FightScene class, representing the scene during a fight between a player and an enemy.

Classes

- class game.FightScene

  Represents the scene during a fight between a player and an enemy.
- enum game.FightScene.FightState

Packages

- package game

9.23.1 Detailed Description

This file contains the implementation of the FightScene class, representing the scene during a fight between a player and an enemy.

Definition in file FightScene.java.

9.24 FightScene.java

Go to the documentation of this file.

```
00001
00006 package game;
00007
00008 import entity.Enemy;
00009 import entity.Player;
00010 import game.Scene.State;
00011
00012 import java.awt.*;
00013 import java.util.HashMap;
00014
00015 import UI.HUD;
00016 import UI.HUD.MenuType;
00017
00018
00024 public class FightScene {
00025     public enum FightState {FIGHTING, WON , LOST};
00026     //private double dt = Scene.getdt();
00027     public Player player;
00028     public Enemy enemy;
00029     public FightState state;
00030     private HUD menu;
00031
00037     public FightScene(Player player, Enemy enemy) {
00038         System.out.println("Entering combat");
00039         this.player = player;
00040         this.enemy = enemy;
00041         state = FightState.FIGHTING;
00042         menu = new HUD(MenuType.FIGHT);
00043     }
00044
00048     public void update(Scene scene) {
00049
00050         // Additional logic for the fight scene update
00051         System.out.println("Le joueur est en combat avec "+ enemy.name);
00052         if(scene.keyH.interactPressed){
00053             state = FightState.WON;
00054             scene.state = State.WORLD;
00055             player.speed = 0;
00056             killEnemy(World.enemies, enemy);
00057
00058             scene.keyH.interactPressed = false;
00059         }
00060     }
00061
00062     public void killEnemy(HashMap<Point, Enemy> enemies, Enemy enemy){
00063         enemies.remove(new Point(enemy.worldX,enemy.worldY), enemy);
00064     }
00065
00066
00073     public void draw(Graphics2D g2) {
00074         g2.setColor( new Color(0xFF2265));
00075         g2.fillRect(100,200,400,150);
00076         player.drawInFight(g2, Const.WDW_width / 2 - (Const.FGHT_entityScreenSize / 2),
      Const.WDW_height / 2 - (Const.FGHT_entityScreenSize / 2));
00077         menu.draw(g2, Const.WDW_width, Const.WDW_height);
00078     }
00079 }
```

9.25 src/game/Scene.java File Reference

This file contains the implementation of the abstract Scene class, representing a scene in the game.

Classes

- class game.Scene

  Represents an abstract scene in the game.
- enum game.Scene.State

Packages

- package game

9.25.1 Detailed Description

This file contains the implementation of the abstract Scene class, representing a scene in the game.

Definition in file Scene.java.

9.26 Scene.java

Go to the documentation of this file.

```
00001
00006 package game;
00007
00008 import main.KeyHandler;
00009
00010 import java.awt.*;
00011
00012 import UI.HUD;
00013 import UI.HUD.MenuType;
00014
00019 public abstract class Scene {
00020
00025     public enum State {WORLD, FIGHT, PAUSE, MENU}
00026
00027     private State _lastState;
00028
00029     public KeyHandler keyH = KeyHandler.getInstance();
00030     protected static double dt = 0;
00031     public State state;
00032     public HUD menu;
00033
00037     public abstract void update();
00038
00045     public abstract void draw(Graphics2D g2, int screenWidth, int screenHeight);
00046
00047
00048     public static double getdt(){
00049         return dt;
00050     }
00051
00056     /*public Scene worldScene() {
00057 return World.getWorld();
00058 }*/
00059
00066     /*public Scene fightScene(Player player, Enemy enemy) {
00067 return new FightScene(player, enemy);
00068 }*/
00069
00070
00074     public void checkPauseScene() {
00075         if (keyH.escPressed) {
00076             keyH.escPressed = false;
00077
00078             if (state != State.PAUSE) {
00079                 System.out.println("CHANGING SCENE TO: PAUSE");
00080                 _lastState = state;
00081                 state = State.PAUSE;
00082                 menu = new HUD(MenuType.PAUSE);
00083             } else {
00084                 System.out.println("CHANGING SCENE TO: "+_lastState);
00085                 state = _lastState;
00086                 menu = null;
00087             }
00088         }
00089     }
00090
00091     public void changeScene(State newState){
00092         System.out.println("CHANGING SCENE TO: " + newState);
00093         this.state = newState;
00094     }
00095
00096 }
```

9.27 src/game/World.java File Reference

This file contains the implementation of the World class, responsible for managing the game world.

Classes

- class game.World

  Represents the game world and manages its entities.

Packages

- package game

9.27.1 Detailed Description

This file contains the implementation of the World class, responsible for managing the game world.

Definition in file World.java.

9.28 World.java

Go to the documentation of this file.

```
00001
00006 package game;
00007
00008 import entity.Player;
00009 import tiles.TileManager;
00010 import entity.Enemy;
00011 import entity.EntitySetter;
00012 import entity.props.Props;
00013 import game.FightScene.FightState;
00014 import java.awt.*;
00015 import java.util.HashMap;
00016 import UI.Textbox;
00017
00018
00024 public class World extends Scene {
00025
00026     private static World _instance;
00027     private FightScene _currfight;
00028     public TileManager tileManager = new TileManager(this);
00029
00030
00031
00032     public EntitySetter entitySetter = new EntitySetter(this); // Instance of EntitySetter
00033
00034     // Doc table de Hashage :  https://www.geeksforgeeks.org/java-util-dictionary-class-java/
00035     public HashMap<Point, Props> objMap = new HashMap<>(); // HashMap to store objects with
      coordinates
00036     public static HashMap<Point, Enemy> enemies = new HashMap<>();
00037
00038
00039     // Player settings
00040     public Player player = new Player("player",15 * Const.WRLD_tileScreenSize,
00041             15 * Const.WRLD_tileScreenSize, 0, 0, 0, "down", 4, 20);
00042
00043
00044     //public Player player;
00050     public static World getWorld() {
00051         Textbox.loadFont("rainyhearts");
00052         if (_instance == null) {
00053             _instance = new World();
00054             _instance.state = State.WORLD;
00055         }
```

```
00056         return _instance;
00057     }
00058 
00059 
00063     public void setupGame() {
00064         entitySetter.setObject();
00065         entitySetter.setEnemies();
00066 
00067         player = new Player("player", 15 * Const.WRLD_tileScreenSize,
00068             15 * Const.WRLD_tileScreenSize, 0, 0, 0, "down", 4, 20);
00069 
00070     }
00071 
00072 
00073 
00080     public void addObject(Point coordinates, Props object) {
00081         objMap.put(coordinates, object);
00082     }
00083 
00084     public void addEnemy(Point coordinates, Enemy enemy){
00085         enemies.put(coordinates,enemy);
00086     }
00087 
00091     public void update() {
00092         checkPauseScene();
00093         if (state == State.WORLD) {
00094             Point ptn = new Point((int)13 * Const.WRLD_entityScreenSize, (int) 13 *
    Const.WRLD_entityScreenSize);
00095             Props key = objMap.get(ptn);
00096             //System.out.println("("+key.worldX +"," + key.worldY + ")  (" + key.hitbox.x
    +","+key.hitbox.y+")");
00097             //int playerScreenX = (Const.WDW_width - Const.WRLD_entityScreenSize) / 2;
00098             //int playerScreenY = (Const.WDW_height - Const.WRLD_entityScreenSize) / 2;
00099 
00100             player.update(this, dt);
00101             tileManager.update(this);
00102 
00103 
00104             //Update enemy if he's close enough otherwise useless to update (5 tile radius around the
    screen)
00105             for(Enemy enemy :  enemies.values()){
00106                 int playerScreenX = (Const.WDW_width - Const.WRLD_entityScreenSize) / 2;
00107                 int playerScreenY = (Const.WDW_height - Const.WRLD_entityScreenSize) / 2;
00108 
00109                 if (enemy.worldX + 5*Const.WRLD_tileScreenSize > player.worldX - playerScreenX
00110                 && enemy.worldX - 5*Const.WRLD_tileScreenSize < player.worldX + playerScreenX
00111                 && enemy.worldY + 5*Const.WRLD_tileScreenSize > player.worldY - playerScreenY
00112                 && enemy.worldY - 5*Const.WRLD_tileScreenSize < player.worldY + playerScreenY) {
00113                     enemy.update(this, dt);
00114 
00115                 }
00116             }
00117         }
00118 
00119 
00120 
00121 
00122         if (state == State.FIGHT){
00123             _currfight.update(this);
00124         }
00125     }
00126 
00134     public void draw(Graphics2D g2, int screenWidth, int screenHeight) {
00135 
00136         //If there is a fight, draw the fight instead of the game world
00137         if(_currfight != null){
00138             if( _currfight.state == FightState.FIGHTING){
00139                 _currfight.draw(g2);
00140             }
00141             if(_currfight.state == FightState.WON){
00142                 _currfight = null;
00143             }
00144         }
00145 
00146         else{
00147             // TILE
00148             tileManager.draw(g2, this);
00149             // OBJECT
00150             for (Props props :  objMap.values()) {
00151                 props.draw(g2, this);
00152             }
00153             // PLAYER
00154             player.drawInWorld(g2, (screenWidth-Const.WRLD_entityScreenSize)/2,
00155                                    (screenHeight-Const.WRLD_entityScreenSize)/2); // Player is always
    centered to screen
00156 
00157 
```

```
00158
00159             //ENEMIES
00160             for(Enemy enemy :  enemies.values()){
00161                 // Calculate the screen position of the character relative to the player's position
00162                 int screenX = enemy.worldX - player.worldX + (Const.WDW_width -
      Const.WRLD_entityScreenSize)/2;
00163                 int screenY = enemy.worldY - player.worldY + (Const.WDW_height -
      Const.WRLD_entityScreenSize)/2;
00164
00165
00166
00167                 enemy.drawInWorld(g2, screenX, screenY);
00168             }
00169         }
00170     }
00171 }
```

9.29 src/item/armor/Armor.java File Reference

Classes

- class item.armor.Armor

Packages

- package item.armor

9.30 Armor.java

Go to the documentation of this file.

```
00001 package item.armor;
00002
00003 import item.Item;
00004
00005 public abstract class Armor extends Item {
00006 }
```

9.31 src/item/armor/Body.java File Reference

Classes

- class item.armor.Body

Packages

- package item.armor

9.32 Body.java

Go to the documentation of this file.

```
00001 package item.armor;
00002
00003 public class Body extends Armor{
00004
00005     public Body(){
00006         System.out.println("Armure body");
00007     }
00008 }
```

9.33 src/item/armor/Foot.java File Reference

Classes

- class item.armor.Foot

Packages

- package item.armor

9.34 Foot.java

Go to the documentation of this file.

```
00001 package item.armor;
00002
00003 public class Foot extends Armor{
00004
00005     public Foot(){
00006         System.out.println("Armure foot");
00007     }
00008 }
```

9.35 src/item/armor/Head.java File Reference

Classes

- class item.armor.Head

Packages

- package item.armor

9.36 Head.java

Go to the documentation of this file.

```
00001 package item.armor;
00002
00003 public class Head extends Armor{
00004     public Head(){
00005         System.out.println("Armure tete");
00006     }
00007 }
```

9.37 src/item/armor/Legs.java File Reference

Classes

- class item.armor.Legs

Packages

- package item.armor

9.38 Legs.java

Go to the documentation of this file.

```
00001 package item.armor;
00002
00003 public class Legs extends Armor{
00004
00005     public Legs(){
00006         System.out.println("Armure jambe");
00007     }
00008 }
```

9.39 src/item/Generator.java File Reference

Classes

- class item.Generator

Packages

- package item

9.40 Generator.java

Go to the documentation of this file.

```
00001 package item;
00002
00003 import item.armor.*;
00004 import item.potion.*;
00005 import item.weapon.*;
00006
00007 public class Generator {
00008
00009     public static Item generateItem(){
00010         int nbRandom= (int) (Math.random() * 3);
00011         return switch (nbRandom) {
00012             case 1 -> Generator.generateWeapon();
00013             case 2 -> Generator.generateArmor();
00014             default -> Generator.generatePotion();
00015         };
00016     }
00017
00018     public static Weapon generateWeapon(){
00019         int nbRandom= (int) (Math.random() * 3);
00020         return switch (nbRandom) {
00021             case 1 -> new Bow();
00022             case 2 -> new Staff();
00023             default -> new Sword();
00024         };
00025     }
00026
00027     public static Potion generatePotion(){
00028         int nbRandom= (int) (Math.random() * 3);
00029         return switch (nbRandom) {
00030             case 1 -> new HealthPotion();
00031             case 2 -> new ManaPotion();
00032             default -> new SpeedPotion();
00033         };
00034     }
```

```
00035
00036     public static Armor generateArmor(){
00037         int nbRandom= (int) (Math.random() * 4);
00038         return switch (nbRandom) {
00039             case 1 -> new Body();
00040             case 2 -> new Head();
00041             case 3 -> new Legs();
00042             default -> new Foot();
00043         };
00044     }
00045 }
```

9.41 src/item/Item.java File Reference

Classes

- class item.Item

Packages

- package item

9.42 Item.java

Go to the documentation of this file.

```
00001 package item;
00002
00003 public abstract class Item {
00004
00005
00006
00007 }
```

9.43 src/item/potion/HealthPotion.java File Reference

Classes

- class item.potion.HealthPotion

Packages

- package item.potion

9.44 HealthPotion.java

Go to the documentation of this file.

```
00001 package item.potion;
00002
00003 public class HealthPotion extends Potion{
00004
00005     public HealthPotion(){
00006         System.out.println("Potion de soin");
00007     }
00008 }
```

9.45 src/item/potion/ManaPotion.java File Reference

Classes

- class item.potion.ManaPotion

Packages

- package item.potion

9.46 ManaPotion.java

Go to the documentation of this file.

```
00001 package item.potion;
00002
00003 public class ManaPotion extends Potion{
00004
00005     public ManaPotion(){
00006         System.out.println("Potion de Mana");
00007     }
00008 }
```

9.47 src/item/potion/Potion.java File Reference

Classes

- class item.potion.Potion

Packages

- package item.potion

9.48 Potion.java

Go to the documentation of this file.

```
00001 package item.potion;
00002
00003 import item.Item;
00004
00005 public abstract class Potion extends Item {
00006 }
```

9.49 src/item/potion/SpeedPotion.java File Reference

Classes

- class item.potion.SpeedPotion

Packages

- package item.potion

9.50 SpeedPotion.java

Go to the documentation of this file.

```
00001 package item.potion;
00002
00003 public class SpeedPotion extends Potion{
00004
00005     public SpeedPotion(){
00006         System.out.println("Potion de vitesse");
00007     }
00008 }
```

9.51 src/item/weapon/Bow.java File Reference

Classes

- class item.weapon.Bow

Packages

- package item.weapon

9.52 Bow.java

Go to the documentation of this file.

```
00001 package item.weapon;
00002
00003 public class Bow extends Weapon{
00004
00005     public Bow(){
00006         System.out.println("Bow");
00007     }
00008 }
```

9.53 src/item/weapon/Staff.java File Reference

Classes

- class item.weapon.Staff

Packages

- package item.weapon

9.54 Staff.java

Go to the documentation of this file.

```
00001 package item.weapon;
00002
00003 public class Staff extends Weapon{
00004
00005     public Staff(){
00006         System.out.println("Staff");
00007     }
00008 }
```

9.55 src/item/weapon/Sword.java File Reference

Classes

- class item.weapon.Sword

Packages

- package item.weapon

9.56 Sword.java

Go to the documentation of this file.

```
00001 package item.weapon;
00002
00003 public class Sword extends Weapon{
00004
00005     public Sword(){
00006         System.out.println("Armure foot");
00007     }
00008 }
```

9.57 src/item/weapon/Weapon.java File Reference

Classes

- class item.weapon.Weapon

Packages

- package item.weapon

9.58 Weapon.java

Go to the documentation of this file.

```
00001 package item.weapon;
00002
00003 import item.Item;
00004
00005 public abstract class Weapon extends Item {
00006
00007
00008 }
```

9.59 src/main/KeyHandler.java File Reference

This file contains the implementation of the KeyHandler class, following the singleton pattern.

Classes

- class main.KeyHandler

  Handles keyboard input using the singleton pattern.

Packages

- package main

9.59.1 Detailed Description

This file contains the implementation of the KeyHandler class, following the singleton pattern.

Definition in file KeyHandler.java.

9.60 KeyHandler.java

Go to the documentation of this file.

```
00001
00006 package main;
00007
00008 import java.awt.event.KeyEvent;
00009 import java.awt.event.KeyListener;
00010
00016 public final class KeyHandler implements KeyListener {
00017     public static KeyHandler instance;
00018     public boolean upPressed, downPressed, leftPressed, rightPressed;
00019     public boolean interactPressed;
00020     public boolean escPressed;
00021
00022     // Private constructor to prevent instantiation from outside the class
00023     private KeyHandler() {}
00024
00030     public static KeyHandler getInstance() {
00031         if (instance == null) {
00032             instance = new KeyHandler();
00033         }
00034         return instance;
00035     }
00036
00037     @Override
00038     public void keyTyped(KeyEvent e) {
00039         // Not used
00040     }
00041
00042     @Override
00043     public void keyPressed(KeyEvent e) {
00044         int code = e.getKeyCode();
00045
00046         if (code == KeyEvent.VK_Z) {
00047             upPressed = true;
00048         }
00049         if (code == KeyEvent.VK_Q) {
00050             leftPressed = true;
00051         }
00052         if (code == KeyEvent.VK_S) {
00053             downPressed = true;
00054         }
00055         if (code == KeyEvent.VK_D) {
00056             rightPressed = true;
```

```
00057         }
00058         if (code == KeyEvent.VK_SPACE) {
00059             interactPressed = true;
00060         }
00061         if (code == KeyEvent.VK_ESCAPE) {
00062             escPressed = true;
00063         }
00064     }
00065
00066     @Override
00067     public void keyReleased(KeyEvent e) {
00068         int code = e.getKeyCode();
00069
00070         if (code == KeyEvent.VK_Z) {
00071             upPressed = false;
00072         }
00073         if (code == KeyEvent.VK_Q) {
00074             leftPressed = false;
00075         }
00076         if (code == KeyEvent.VK_S) {
00077             downPressed = false;
00078         }
00079         if (code == KeyEvent.VK_D) {
00080             rightPressed = false;
00081         }
00082         if (code == KeyEvent.VK_SPACE) {
00083             interactPressed = false;
00084         }
00085         if (code == KeyEvent.VK_ESCAPE) {
00086             escPressed = false;
00087         }
00088     }
00089 }
```

9.61 src/tiles/Tile.java File Reference

This file contains the implementation of the Tile class, which represents a tile in the game world.

Classes

- class tiles.Tile

  Represents a tile in the game world.

Packages

- package tiles

9.61.1 Detailed Description

This file contains the implementation of the Tile class, which represents a tile in the game world.

Definition in file Tile.java.

9.62 Tile.java

Go to the documentation of this file.

```
00001
00006 package tiles;
00007
00008 import java.awt.Graphics2D;
00009 import java.awt.image.BufferedImage;
00010 import java.io.FileInputStream;
00011 import java.io.IOException;
00012
00013 import javax.imageio.ImageIO;
00014
00015 import game.Const;
00016
00021 public class Tile {
00022     private boolean _isBlocking = false;
00023     private int _worldX, _worldY;
00024
00025     // Display purpose variables
00026     private final int tileSize = 16;
00027     private final int scale = 3;
00028     public final int screenSize = tileSize * scale;
00029
00030     private int _spriteCnt = 0; // Variable responsible for the incrementation of the different
      sprites
00031     private int _spriteUpdater = 0; // Variable responsible for the incrementation of the speed of the
      sprites
00032     public int spriteSpeed; // How fast are the sprites changing (higher spriteSpeed means slower time
      to change)
00033     // spriteSpeed of 0 means it has NO animation
00034     public int spriteCntMax; // How many sprite does the entity have
00035
00036     public BufferedImage[] image;
00037
00046     public Tile(int spriteCntMax, int spriteSpeed, boolean isBlocking, int ind) {
00047         this.spriteCntMax = spriteCntMax;
00048         this.spriteSpeed = spriteSpeed;
00049         this._isBlocking = isBlocking;
00050         image = new BufferedImage[spriteCntMax];
00051     }
00052
00058     public void setCollision(boolean collision) {
00059         _isBlocking = collision;
00060     }
00061
00067     public void setTexture(BufferedImage[] newTexture) {
00068         this.image = newTexture;
00069     }
00070
00077     public void setPos(int x, int y) {
00078         this._worldX = x;
00079         this._worldY = y;
00080     }
00081
00087     public int[] getPos() {
00088         int tmp[] = new int[2];
00089         tmp[0] = _worldX;
00090         tmp[1] = _worldY;
00091         return tmp;
00092     }
00093
00099     public boolean getCollision() {
00100         return _isBlocking;
00101     }
00102
00109     public void setSpriteCountAndSpeed(int newSpriteCntMax, int newSpriteSpeed) {
00110         this.spriteCntMax = newSpriteCntMax;
00111         this.spriteSpeed = newSpriteSpeed;
00112     }
00113
00121     public void loadTextures(String name, boolean animated, int i) {
00122         try {
00123             if (animated) {
00124                 for (int j = 0; j < spriteCntMax; j++) {
00125                     image[j] = ImageIO.read(new FileInputStream("res/tiles/animated/" + name + (j + 1)
      + ".png"));
00126                 }
00127             } else {
00128                 image[0] = ImageIO.read(new FileInputStream("res/tiles/static/" + name + (i + 1) +
      ".png"));
00129             }
00130         } catch (IOException e) {
00131             e.printStackTrace();
00132         }
```

```
00133     }
00134
00138     public void updateFrames() {
00139         if (spriteSpeed > 0) {
00140             _spriteUpdater++;
00141             if (_spriteUpdater > spriteSpeed) {
00142                 _spriteCnt++;
00143                 if (_spriteCnt == spriteCntMax) {
00144                     _spriteCnt = 0;
00145                 }
00146                 _spriteUpdater = 1;
00147             }
00148         }
00149     }
00150
00158     public void draw(Graphics2D g2, int screenX, int screenY) {
00159         BufferedImage render = null;
00160
00161         for (int i = 0; i < spriteCntMax; i++) {
00162             if (_spriteCnt == i) {
00163                 render = image[i];
00164             }
00165         }
00166
00167         g2.drawImage(render, screenX, screenY, screenSize, screenSize, null);
00168         //g2.drawRect(screenX,screenY,Const.WRLD_tileScreenSize,Const.WRLD_tileScreenSize);
      //Debugging purposes
00169     }
00170
00171 }
```

9.63 src/tiles/TileManager.java File Reference

This file contains the implementation of the TileManager class, which manages tiles in the game world.

Classes

- class tiles.TileManager

  Manages tiles in the game world.

Packages

- package tiles

9.63.1 Detailed Description

This file contains the implementation of the TileManager class, which manages tiles in the game world.

Definition in file TileManager.java.

9.64 TileManager.java

Go to the documentation of this file.

```
00001
00006 package tiles;
00007
00008 import game.Const;
00009 import game.World;
00010
00011 import java.awt.Graphics2D;
00012 import java.io.BufferedReader;
00013 import java.io.File;
00014 import java.io.FileReader;
00015
00020 public class TileManager {
00021
00022
00023     private Tile[] _floorMapTextures;
00024     private Tile[] _topMapTextures;
00025
00026     private Tile[][] _floorMap;
00027     private Tile[][] _topMap;
00028
00034     public TileManager(World world) {
00035         // Textures
00036         _floorMapTextures = new Tile[Const.nbFloorTextures];
00037         _topMapTextures = new Tile[Const.nbTopTextures];
00038
00039         // Map itself
00040         _floorMap = new Tile[Const.WRLD_maxRow][Const.WRLD_maxCol];
00041         _topMap = new Tile[Const.WRLD_maxRow][Const.WRLD_maxCol];
00042
00043         loadTextures();
00044         loadMap("res/maps/debugfloor.txt", world, _floorMap, _floorMapTextures);
00045         loadMap("res/maps/debugtop.txt", world, _topMap, _topMapTextures);
00046     }
00047
00055     public Tile getTile(int x, int y) {
00056         int param = Const.WRLD_tileScreenSize;
00057         int i = x / param;
00058         int j = y / param;
00059
00060         return _topMap[j][i];
00061     }
00062
00075     private void storeTexture(String name, Tile[] tiles, int start, int size, boolean animated, int
      spriteCntMax,
00076                               int spriteSpeed, boolean isBlocking) {
00077         for (int i = start; i < size + start; i++) {
00078             Tile tile = new Tile(spriteCntMax, spriteSpeed, isBlocking, i);
00079             tiles[i] = tile;
00080             tiles[i].loadTextures(name, animated, i - start);
00081         }
00082     }
00083
00087     private void loadTextures() {
00088         // Floor map textures
00089         storeTexture("grass", _floorMapTextures, 0, 3, false, 1, 0, false);
00090         // Decoration textures (plants and tall grass)
00091         storeTexture("grass0", _floorMapTextures, 3, 1, true, 5, 20, false);
00092         storeTexture("grass1", _floorMapTextures, 4, 1, true, 5, 20, false);
00093         storeTexture("flower0", _floorMapTextures, 5, 1, true, 6, 20, false);
00094
00095         // Top map textures (trees & forest)
00096         storeTexture("void", _topMapTextures, 0, 1, false, 1, 0, false);
00097
00098         storeTexture("forest", _topMapTextures, 1, 9, false, 1, 0, true);
00099         storeTexture("tree", _topMapTextures, 10, 9, false, 1, 0, true);
00100
00101         storeTexture("forest_topleft", _topMapTextures, 19, 3, false, 1, 0, true);
00102         storeTexture("forest_topright", _topMapTextures, 22, 2, false, 1, 0, true);
00103         storeTexture("forest_bottomleft", _topMapTextures, 24, 2, false, 1, 0, true);
00104         storeTexture("forest_bottomright", _topMapTextures, 26, 1, false, 1, 0, true);
00105         storeTexture("fire", _topMapTextures, 27, 1, true, 7, 15, true);
00106     }
00107
00116     private void loadMap(String filePath, World world, Tile[][] mapTile, Tile[] textures) {
00117         try {
00118             File file = new File(filePath);
00119             FileReader fileReader = new FileReader(file);
00120             BufferedReader br = new BufferedReader(fileReader);
00121
00122             for (int i = 0; i < Const.WRLD_maxRow; i++) {
00123                 String line = br.readLine();
00124
```

```
00125                 for (int j = 0; j < Const.WRLD_maxCol; j++) {
00126                     String[] numbers = line.split("\\s+");
00127                     int num = Integer.parseInt(numbers[j]); // Reading the file itself and stocking
      int read
00128 
00129                     Tile tileCurrent = new Tile(textures[num].spriteCntMax, textures[num].spriteSpeed,
00130                             textures[num].getCollision(), num); // Creating new tile to store with the
      according texture
00131                     tileCurrent.setPos(Const.WRLD_tileScreenSize * j, Const.WRLD_tileScreenSize * i);
00132 
00133                     mapTile[i][j] = tileCurrent; // Setting the tile to the actual map
00134                     mapTile[i][j].setTexture(textures[num].image); // Set the current tile texture to
      what corresponds
00135                     // in the textures array
00136                     mapTile[i][j].setCollision(textures[num].getCollision()); // Set the collision
      factor to the tile
00137 
00138                     // System.out.print(num + " ");
00139                 }
00140                 // System.out.println("");
00141             }
00142             br.close();
00143 
00144         } catch (Exception e) {
00145             e.printStackTrace();
00146         }
00147     }
00148 
00156     public void update(World world) {
00157         int playerScreenX = (Const.WDW_width - Const.WRLD_entityScreenSize) / 2;
00158         int playerScreenY = (Const.WDW_height - Const.WRLD_entityScreenSize) / 2;
00159 
00160         for (int i = 0; i < Const.WRLD_maxRow; i++) {
00161             for (int j = 0; j < Const.WRLD_maxCol; j++) {
00162                 int worldX = _floorMap[i][j].getPos()[0];
00163                 int worldY = _floorMap[i][j].getPos()[1];
00164 
00165                 // Checks if the player is close enough to the tile to render it to optimize memory
      and CPU usage
00166                 if (worldX + Const.WRLD_tileScreenSize > world.player.worldX - playerScreenX
00167                         && worldX - Const.WRLD_tileScreenSize < world.player.worldX + playerScreenX
00168                         && worldY + Const.WRLD_tileScreenSize > world.player.worldY - playerScreenY
00169                         && worldY - Const.WRLD_tileScreenSize < world.player.worldY + playerScreenY) {
00170                     _floorMap[i][j].updateFrames();
00171                     _topMap[i][j].updateFrames();
00172                 }
00173             }
00174         }
00175     }
00176 
00185     public void draw(Graphics2D g2, World world) {
00186         int playerScreenX = (Const.WDW_width - Const.WRLD_entityScreenSize) / 2;
00187         int playerScreenY = (Const.WDW_height - Const.WRLD_entityScreenSize) / 2;
00188 
00189         // Check for every tile of the map if it needs to be drawn
00190         for (int i = 0; i < Const.WRLD_maxRow; i++) {
00191             for (int j = 0; j < Const.WRLD_maxCol; j++) {
00192                 int worldX = _floorMap[i][j].getPos()[0];
00193                 int worldY = _floorMap[i][j].getPos()[1];
00194 
00195                 // Checks if the player is close enough to the tile to render it to optimize memory
      and CPU usage
00196                 if (worldX + Const.WRLD_tileScreenSize > world.player.worldX - playerScreenX
00197                         && worldX - Const.WRLD_tileScreenSize < world.player.worldX + playerScreenX
00198                         && worldY + Const.WRLD_tileScreenSize > world.player.worldY - playerScreenY
00199                         && worldY - Const.WRLD_tileScreenSize < world.player.worldY + playerScreenY) {
00200                     int screenX = worldX - world.player.worldX + playerScreenX;
00201                     int screenY = worldY - world.player.worldY + playerScreenY;
00202                     _floorMap[i][j].draw(g2, screenX, screenY);
00203                     _topMap[i][j].draw(g2, screenX, screenY);
00204                 }
00205             }
00206         }
00207     }
00208 }
```

9.65 src/UI/ChoiceButton.java File Reference

Classes

- class UI.ChoiceButton

Packages

- package UI

9.66 ChoiceButton.java

Go to the documentation of this file.

```
00001 package UI;
00002
00003
00004 import java.awt.image.BufferedImage;
00005 import java.io.FileInputStream;
00006 import java.io.IOException;
00007
00008 import javax.imageio.ImageIO;
00009
00010 import java.awt.Graphics2D;
00011 import java.awt.Color;
00012
00013 public class ChoiceButton {
00014     public int width, height;
00015     private Textbox _textBox;
00016     private BufferedImage _bgTexture;
00017
00018     public ChoiceButton(int w,int h, String title, String fontName, Color fontColor){
00019         this.width = w; this.height = h;
00020
00021         _textBox = new Textbox(title,fontName,w,h,fontColor);
00022
00023         loadTexture();
00024     }
00025
00026     private void loadTexture(){
00027         try{
00028             _bgTexture = ImageIO.read(new FileInputStream("res/hud/bg.png"));
00029         }
00030         catch(IOException e){
00031             e.printStackTrace();
00032         }
00033     }
00034
00035     public void draw(Graphics2D g2, int x, int y){
00036         //g2.drawImage(_bgTexture,x,y,80*3,20*3,null);
00037         g2.setColor(new Color(0xA38168));
00038         g2.fillRect(x,y,width,height);
00039         _textBox.draw(g2,x,y);
00040     }
00041 }
```

9.67 src/UI/HUD.java File Reference

Classes

- class UI.HUD
- enum UI.HUD.MenuType

Packages

- package UI

9.68 HUD.java

Go to the documentation of this file.

```
00001 package UI;
00002
00003 import java.awt.Color;
00004 import java.awt.Graphics2D;
00005
00006 public class HUD {
00007     public enum MenuType {WELCOME, PAUSE, FIGHT, SHOP}
00008
00009     private int _nbButtons;
00010     private ChoiceButton[] _buttons;
00011     private ChoiceButton _title;
00012     private MenuType _type;
00013     private int _titleWidth, _titleHeight;
00014     //private KeyHandler keyH = KeyHandler.getInstance();
00015
00016
00017     public final int HUDWidth = 600;
00018     public final int HUDHeight = 550;
00019
00020     public HUD(MenuType type){
00021         _type = type;
00022         switch(_type){
00023             case WELCOME:
00024                 _nbButtons = 2;
00025                 break;
00026             case PAUSE:
00027                 _nbButtons = 3;
00028                 _titleWidth = 300;
00029                 _titleHeight = 80;
00030                 _title = new ChoiceButton(_titleWidth,_titleHeight,"PAUSE","rainyhearts",new
      Color(0x834317));
00031                 break;
00032             case FIGHT:
00033                 _titleWidth = 300;
00034                 _titleHeight = 60;
00035                 _nbButtons = 4;
00036                 _title = new ChoiceButton(_titleWidth,_titleHeight,"FIGHT","rainyhearts",new
      Color(0xF10516));
00037                 break;
00038             case SHOP:
00039                 _nbButtons = 4;
00040                 break;
00041         }
00042
00043         _buttons = new ChoiceButton[_nbButtons];
00044
00045         //To replace with the current names that we want depending on the MenuType
00046
00047         for(int i =0; i<_nbButtons ; i++){
00048             _buttons[i] = new ChoiceButton(80*3,20*3, "BUTTON " +i, "rainyhearts",Color.black);
00049         }
00050     }
00051
00052
00053     public void draw(Graphics2D g2, int screenWidth, int screenHeight){
00054         //g2.drawImage(_bgTexture, (800 - HUDWidth)/2,(600 - HUDHeight)/2,HUDWidth, HUDHeight, null);
00055         g2.setColor(new Color(0,0,0,20));
00056         g2.fillRect((screenWidth - HUDWidth)/2, (screenHeight - HUDHeight)/2,HUDWidth,HUDHeight);
      //"Drawing" HUD with soft background color
00057
00058
00059
00060
00061         int gap = (HUDHeight - _nbButtons*_buttons[0].height)/(_nbButtons+1);    //Small gap to space
      the buttons and keep them centered on screen
00062         _title.draw(g2,(screenWidth - _titleWidth)/2, screenHeight - HUDHeight);
00063         for(int i=0;i < _nbButtons; i++){
00064             _buttons[i].draw(g2,(screenWidth - _buttons[i].width)/2, (gap+gap*(i+1)/2 +
      _buttons[i].height*i + (screenHeight - HUDHeight)/2));
00065         }
00066     }
00067
00068     public void update(double dt){
00069
00070     }
00071 }
```

9.69 src/UI/Textbox.java File Reference

Classes

- class UI.Textbox

Packages

- package UI

9.70 Textbox.java

Go to the documentation of this file.

```
00001 package UI;
00002
00003 import java.awt.Color;
00004 import java.awt.Font;
00005 import java.awt.FontFormatException;
00006 import java.awt.Graphics2D;
00007 import java.awt.GraphicsEnvironment;
00008 import java.io.File;
00009 import java.io.IOException;
00010
00011 import javax.swing.JLabel;
00012 public class Textbox extends JLabel{
00013     private int _width, _height;
00014     private Color _color;
00015     private String _text;
00016     private Font _font;
00017     private int _fontSizeToUse;
00018
00019
00020     public Textbox(String text, String fontName, int w, int h, Color color){
00021         _width = w; _height = h;
00022         _text = text;
00023         _color = color;
00024
00025         //loadFont(fontName);
00026         _font = new Font(fontName, Font.PLAIN, 1);
00027
00028
00029         int stringWidth = this.getFontMetrics(_font).stringWidth(_text);
00030         int componentWidth = _width;
00031
00032         // Find out how much the font can grow in width.
00033         double widthRatio = (double)componentWidth / (double)stringWidth;
00034
00035         int newFontSize = (int)(_font.getSize() * widthRatio);
00036         int componentHeight = _height;
00037
00038         // Pick a new font size so it will not be larger than the height of label.
00039         _fontSizeToUse = Math.min(newFontSize, componentHeight);
00040
00041         // Set the label's font size to the newly determined size.
00042         _font = new Font(fontName, Font.PLAIN, _fontSizeToUse);
00043     }
00044
00045
00046     public static void loadFont(String fontName){
00047         try {
00048             GraphicsEnvironment ge = GraphicsEnvironment.getLocalGraphicsEnvironment();
00049             ge.registerFont(Font.createFont(Font.TRUETYPE_FONT, new
      File("res/hud/font/rainyhearts.ttf")));
00050         }
00051         catch (FontFormatException e) {
00052             e.printStackTrace();
00053         }
00054         catch (IOException e) {
00055             e.printStackTrace();
00056         }
00057     }
00058
00059     public void draw(Graphics2D g2,int x, int y){
00060         //TODO: find a way to center text on the button
```

```
00061         g2.setFont(_font);
00062         g2.setColor(Color.black);
00063         g2.drawRect(x,y,_width,_height);
00064         g2.setColor(_color);
00065         g2.drawString(_text, x + _fontSizeToUse, y + (_height +_fontSizeToUse)/2);
00066     }
00067
00068 }
```

9.71 test/CreationCombatScene/entity/PlayerTest.java File Reference

Classes

- class CreationCombatScene.entity.PlayerTest

Packages

- package CreationCombatScene.entity

9.72 PlayerTest.java

Go to the documentation of this file.

```
00001 package CreationCombatScene.entity;
00002
00003 import entity.Player;
00004
00005 public class PlayerTest extends Player {
00006
00007
00008     public PlayerTest(){
00009         super(0,0,0,0,0," ",0,0);
00010     }
00011
00012     public  void  update(double dt){
00013
00014         System.out.println("Je suis modifié");
00015
00016     }
00017 }
```

9.73 src/game/Window.java File Reference

This file contains the implementation of the Window class, responsible for displaying the game based on the backend (World.java).

Classes

- class game.Window

  Represents the window that displays the game.

Packages

- package game

9.73.1 Detailed Description

This file contains the implementation of the Window class, responsible for displaying the game based on the backend (World.java).

Definition in file Window.java.

9.74 Window.java

Go to the documentation of this file.

```
00001
00006 package game;
00007
00008 import javax.swing.JPanel;
00009
00010 import game.Scene.State;
00011
00012 import java.awt.*;
00013
00020 public class Window extends JPanel implements Runnable {
00024     public Scene scene;
00025
00029     public final int screenWidth = 800;
00030
00034     public final int screenHeight = 600;
00035
00039     World world = World.getWorld();
00040
00044     private final int _FPS = 60;
00045
00049     Thread gameThread;
00050
00054     public Window() {
00055         scene = World.getWorld();
00056         this.setPreferredSize(new Dimension(screenWidth, screenHeight));
00057         this.setBackground(Color.black);
00058         this.setDoubleBuffered(true);
00059         this.addKeyListener(scene.keyH);
00060         this.setFocusable(true);
00061         world.setupGame(); // Creation of instance setter
00062     }
00063
00067     @Override
00068     public void run() {
00069         double drawInterval = 10e9 / _FPS;
00070         long lastTime = System.nanoTime();
00071         long currentTime;
00072
00073         while (gameThread != null) {
00074             currentTime = System.nanoTime();
00075
00076             Scene.dt += (currentTime - lastTime) / drawInterval;
00077             lastTime = currentTime;
00078
00079             if (Scene.dt > 0.1) {
00080                 update(); // update() method for window both updates the world and repaints the
      components of it
00081                 Scene.dt -= 0.05;
00082             }
00083         }
00084     }
00085
00089     public void startGameThread() {
00090         gameThread = new Thread(this);
00091         gameThread.start();
00092     }
00093
00097     public void update() {
00098         scene.update(); // Updates the whole world's props & animations
00099         repaint();
00100     }
00101
00106     @Override
00107     public void paintComponent(Graphics g) {
00108         super.paintComponent(g);
00109
00110         Graphics2D g2 = (Graphics2D) g;
00111
```

```
00112         scene.draw(g2, screenWidth, screenHeight);
00113
00114         // Darkens the scene on the background to let the menu on top
00115         if (scene.state == State.PAUSE) {
00116             g2.setColor(new Color(0, 0, 0, 180));
00117             g2.fillRect(0, 0, screenWidth, screenHeight);
00118             scene.menu.draw(g2, screenWidth, screenHeight);
00119         }
00120
00121         g2.dispose();
00122     }
00123 }
```

9.75 test/CreationCombatScene/game/Window.java File Reference

Classes

- class CreationCombatScene.game.Window

Packages

- package CreationCombatScene.game

9.76 Window.java

Go to the documentation of this file.

```
00001 package CreationCombatScene.game;
00002
00003 import CreationCombatScene.entity.PlayerTest;
00004 import entity.Enemy;
00005 import game.Scene;
00006
00007 import javax.swing.*;
00008 import java.awt.*;
00009
00010 /*
00011 * This class's purpose is to keep track of the displaying of the game, based on what is on the backend
      (World.java)
00012 */
00013 public class Window extends JPanel implements Runnable{
00014     //Game world
00015     public Scene scene;
00016
00017     //Game screen
00018     public final int screenWidth=800;
00019     public final int screenHeight=600;
00020
00021
00022     //Game init
00023     final int FPS = 120;
00024     Thread gameThread;
00025
00026
00027
00028     public Window(){
00029         PlayerTest playerTest=new PlayerTest();
00030         Enemy ennemy=new Enemy();
00031
00032         scene = scene.fightScene(playerTest,ennemy);
00033         this.setPreferredSize(new Dimension(screenWidth,screenHeight));
00034         this.setBackground(Color.black);
00035         this.setDoubleBuffered(true);
00036         this.addKeyListener(scene.keyH);
00037         this.setFocusable(true);
00038
00039
00040     }
00041
00042     @Override
00043     public void run() {
```

```
00044         double drawInterval = 10e9/FPS;
00045         long lastTime = System.nanoTime();
00046         long currentTime;
00047 
00048         while(gameThread != null){
00049             currentTime = System.nanoTime();
00050 
00051             scene.dt += (currentTime - lastTime) / drawInterval;
00052             lastTime = currentTime;
00053 
00054             if(scene.dt > 0.1){
00055                 update();           //update() method for window both updates the world and repaints
      the components of it
00056                 scene.dt-= 0.1;
00057             }
00058         }
00059     }
00060 
00061     public void startGameThread(){
00062         gameThread = new Thread(this);
00063         gameThread.start();
00064     }
00065 
00066     public void update(){
00067         scene.update();         //Updating abstract class scene which means either world or fightscene
00068         repaint();
00069     }
00070 
00071 
00072     public void paintComponent(Graphics g){
00073 
00074         super.paintComponent(g);
00075 
00076         Graphics2D g2= (Graphics2D)g;
00077 
00078         scene.draw(g2,screenWidth,screenHeight);
00079 
00080         g2.dispose();
00081     }
00082 }
```

9.77 src/main/Main.java File Reference

This file contains the implementation of the Main class, which contains the main method to start the 2D Adventure game.

Classes

- class main.Main

  Contains the main method to start the 2D Adventure game.

Packages

- package main

9.77.1 Detailed Description

This file contains the implementation of the Main class, which contains the main method to start the 2D Adventure game.

Definition in file Main.java.

9.78 Main.java

Go to the documentation of this file.

```
00001
00006 package main;
00007
00008 import javax.swing.*;
00009
00010 import game.Window;
00011
00016 public class Main {
00022     public static void main(String[] args) {
00023         // Create a JFrame (window) for the game
00024         JFrame windows = new JFrame();
00025         windows.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
00026         windows.setResizable(false);
00027         windows.setTitle("2D Adventure");
00028
00029         // Create an instance of the Window class (game window)
00030         Window gameWindow = new Window();
00031         windows.add(gameWindow);
00032
00033         // Pack the components of the window
00034         windows.pack();
00035
00036         // Set the window to appear at the center of the screen
00037         windows.setLocationRelativeTo(null);
00038
00039         // Make the window visible
00040         windows.setVisible(true);
00041
00042         // Start the game thread to handle game logic and rendering
00043         gameWindow.startGameThread();
00044     }
00045 }
```

9.79 test/CreationCombatScene/Main.java File Reference

Classes

- class CreationCombatScene.Main

Packages

- package CreationCombatScene

9.80 Main.java

Go to the documentation of this file.

```
00001 package CreationCombatScene;
00002
00003 import CreationCombatScene.game.Window;
00004
00005 import javax.swing.*;
00006
00007 // Press Shift twice to open the Search Everywhere dialog and type `show whitespaces`,
00008 // then press Enter.  You can now see whitespace characters in your code.
00009 public class Main {
00010     public static void main(String[] args) {
00011
00012         JFrame windows = new JFrame();
00013         windows.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
00014         windows.setResizable(false);
00015         windows.setTitle("2D Adventure");
00016
00017         Window gameWindow = new Window();
00018         windows.add(gameWindow);
00019
```

```
00020         windows.pack();
00021
00022         windows.setLocationRelativeTo(null);
00023         windows.setVisible(true);
00024
00025         gameWindow.startGameThread();
00026     }
00027 }
```

Index