Philophobia

Generated by Doxygen 1.8.6

Mon Apr 7 2014 17:21:49

Contents

Chapter 1

Philophobia

![Build Status](https://drone.io/github.com/minijackson/Philophobia/status.-png)

Sadistic Java game project

Chapter 2

Namespace Index

2.1 Packages

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

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

Chapter 4

Class Index

4.1 Class List

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

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

Chapter 6

Namespace Documentation

6.1 Package debug

Classes

- class CliOptions

  Class used to analyze the commands passed to the program.
- class Verbose

  Class used to display and log messages all over this program.

6.2 Package gameplay

Packages

- package ai

Classes

- class GamePlay

  Class handling the game play of this game.

6.3 Package gameplay.ai

Packages

- package mood
- package phrasing

Classes

- class AI

  Class representing the sadistic robot.

6.4 Package gameplay.ai.mood

Classes

- class Anger
- class Curiosity
- class Depression
- class Mood

  Abstract class representing an AI mood.
- class ProbabilityMood

  Class used to associate a mood with a probability.
- class PowerComplex

6.5 Package gameplay.ai.phrasing

Classes

- class OrderedPhrases

  Class used to contain an ordered list of phrases.
- class Phrasing

6.6 Package main

Classes

- class Philophobia

  Main class.

6.7 Package window

Packages

- package topbar
- package ui

Classes

- class Window
- enum WindowState

  Enumeration of the possible states of the displayed graphics.

6.8 Package window.topbar

Classes

- class TopBar

  Class handling the top bar of the program's main window.
- class TopBarButton

  Class handling the UI buttons.

6.9 Package window.ui

Classes

- class UI

  Class used to handle the window's user interface.

6.10 Package world

Packages

- package character
- package scenery

Classes

- interface InteractiveObject
- class World

  World is a class used to handle the game world graphics.

6.11 Package world.character

Classes

- class Character

  Class used to handle a character (player or non-player)
- class Hero
- class TalkingCharacter

  Class handling a talking character.

6.12 Package world.scenery

Classes

- interface AlpineTundraTheme

  Interface used to define that a Scenery object have an Alpine Tundra style.
- interface CaveTheme

  Interface used to tell that a Scenery object has a Cave style.
- class Flower

  Class used to handle a flower object.
- class Grass

  Class used to handle a grass object.
- class Ground

  Class used to handle a ground object.
- interface PolarDesertTheme

  Interface used to define that a Scenery object has a Polar Desert style.
- class Rock

  Class used to handle a rock object.

- interface SavannaTheme

    Interface used to define that a Scenery object have a Savanna style.

- class Scenery

    Class used to handle any world object.

- class Shore

    Class used to handle a shore object.

- class Shrub

    Class used to handle a shrub object.

- interface SteppeTheme

    Interface used to define that a Scenery object have a Steppe style.

- interface TaigaTheme

    Interface used to define that a Scenery object have a Taiga style.

- interface TemperateBroadleafTheme

    Interface used to define that a Scenery object have a Temperate Broadleaf style.

- class Tree

    Class used to handle a tree object.

- class Water

    Class used to handle a water object.

- interface XericShrublandsTheme

    Interface used to define that a Scenery object have a Xeric Shrublands style.

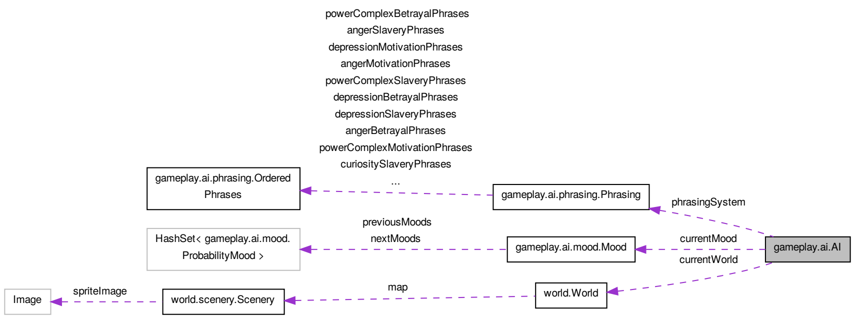
Chapter 7

Class Documentation

7.1 gameplay.ai.AI Class Reference

Class representing the sadistic robot.

Collaboration diagram for gameplay.ai.AI:



Public Member Functions

- AI (World currentWorld)

  AI class main constructor.
- void setCurrentWorld (World world)

  Setter for the current world field.
- void newBetrayal ()

  Method called when the player make a new betrayal (does not do what the AI asked)
- void newSlavery ()

  Method called when the player do what the AI asked.

Protected Member Functions

- void changeMood (Class< Mood > moodClass)

  Change the current mood field from a Class object.

Protected Attributes

- World currentWorld

  World where the player is currently in.
- Mood currentMood

  Mood in which the AI is currently in.
- Phrasing phrasingSystem

  AI talk system.
- int betrayalCount

  Number of betrayals done by the player.
- int slaveryCount

  Number of asked actions by the AI done by the player.

7.1.1 Detailed Description

Class representing the sadistic robot.

The robot ask the player to do terrible things without taking out the player's choice to do or not to do the task

The AI has a mood system with 4 basic moods : Curiosity, Anger, Depression, Power complex

The robot can switch between these moods considering the user's choices and interact with the environment in a bad way

See Also

gameplay.ai.mood.Mood

Definition at line 28 of file AI.java.

7.1.2 Constructor & Destructor Documentation

7.1.2.1 gameplay.ai.AI.AI (World *currentWorld*)

AI class main constructor.

Parameters

currentWorld	World where the player is currently in

Definition at line 60 of file AI.java.

```
                                {
        Philophobia.getVerbose().information("Creating sadistic AI", "gameplay/ai/AI.java", "AI.AI(World)")
    ;

        this.currentWorld = currentWorld;
        currentMood  = new Curiosity();
        phrasingSystem = new Phrasing();
        betrayalCount = 0;
        slaveryCount = 0;
    }
```

7.1.3 Member Function Documentation

7.1.3.1 void gameplay.ai.AI.changeMood (Class< Mood > *moodClass*) [protected]

Change the current mood field from a Class object.

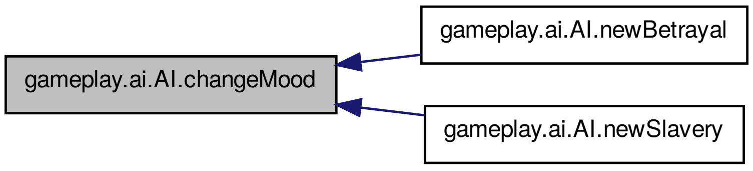
See Also

currentMood

Definition at line 112 of file AI.java.

```
                                                     {
        try {
            Philophobia.getVerbose().information("AI switching from " +
      currentMood.getClass().getName() + " mood to " + moodClass.getName() + " mood", "
      gameplay/ai/AI.java", "AI.changeMood(Class)");
            currentMood = moodClass.newInstance();
        } catch(SecurityException e) {
            Philophobia.getVerbose().serious("Security exception when switching from " +
      currentMood.getClass().getName() + " mood to " + moodClass.getName() + " mood: " + e.getMessage(
      ), "gameplay/ai/AI.java", "AI.changeMood(Class)");
        } catch(InstantiationException e) {
            Philophobia.getVerbose().serious("Instantiation exception when switching from " +
      currentMood.getClass().getName() + " mood to " + moodClass.getName() + " mood: " + e.getMessage(
      ), "gameplay/ai/AI.java", "AI.changeMood(Class)");
        } catch(IllegalAccessException e) {
            Philophobia.getVerbose().serious("Illegal access exception when switching from " +
      currentMood.getClass().getName() + " mood to " + moodClass.getName() + " mood: " + e.getMessage(
      ), "gameplay/ai/AI.java", "AI.changeMood(Class)");
        }
    }
```

Here is the caller graph for this function:


gameplay.ai.AI.changeMood
gameplay.ai.AI.newBetrayal
gameplay.ai.AI.newSlavery

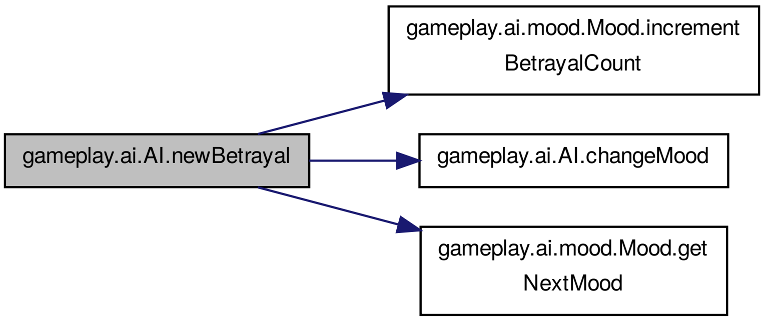
7.1.3.2 void gameplay.ai.AI.newBetrayal ( )

Method called when the player make a new betrayal (does not do what the AI asked)

Definition at line 87 of file AI.java.

```
                              {
        Philophobia.getVerbose().calls("New betrayal action detected", "gameplay/ai/AI.java", "
      AI.newBetrayal()");
        betrayalCount++;
        if(currentMood.incrementBetrayalCount() >=
      currentMood.getBetrayalThreshold()) {
            changeMood(currentMood.getNextMood());
        }
    }
```

Here is the call graph for this function:


gameplay.ai.mood.Mood.increment BetrayalCount
gameplay.ai.AI.newBetrayal
gameplay.ai.AI.changeMood
gameplay.ai.mood.Mood.get NextMood

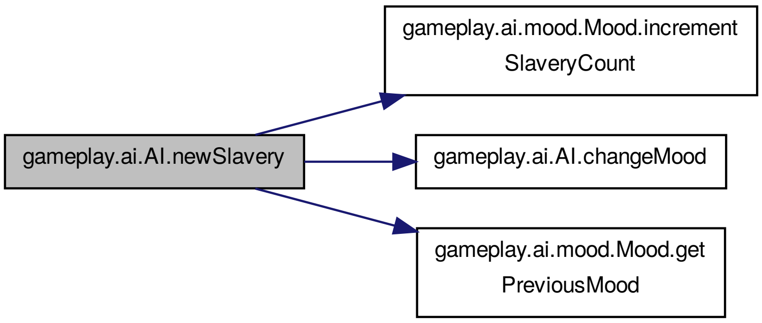
7.1.3.3 void gameplay.ai.AI.newSlavery ( )

Method called when the player do what the AI asked.

Definition at line 99 of file AI.java.

```
                            {
        Philophobia.getVerbose().calls("New slavery action detected", "gameplay/ai/AI.java", "
  AI.newSlavery()");
        slaveryCount++;
        if(currentMood.incrementSlaveryCount() >=
  currentMood.getSlaveryThreshold()) {
            changeMood(currentMood.getPreviousMood());
        }
    }
```

Here is the call graph for this function:


gameplay.ai.mood.Mood.increment SlaveryCount
gameplay.ai.AI.newSlavery
gameplay.ai.AI.changeMood
gameplay.ai.mood.Mood.get PreviousMood

7.1.3.4 void gameplay.ai.AI.setCurrentWorld (World *world*)

Setter for the current world field.

See Also

currentWorld

Definition at line 74 of file AI.java.

```
                                                  {
            currentWorld = world;
        }
```

7.1.4 Member Data Documentation

7.1.4.1 int gameplay.ai.AI.betrayalCount [protected]

Number of betrayals done by the player.

Definition at line 48 of file AI.java.

7.1.4.2 Mood gameplay.ai.AI.currentMood [protected]

Mood in which the AI is currently in.

Definition at line 38 of file AI.java.

7.1.4.3 World gameplay.ai.AI.currentWorld [protected]

World where the player is currently in.

Definition at line 33 of file AI.java.

7.1.4.4 Phrasing gameplay.ai.AI.phrasingSystem [protected]

AI talk system.

Definition at line 43 of file AI.java.

7.1.4.5 int gameplay.ai.AI.slaveryCount [protected]

Number of asked actions by the AI done by the player.

Definition at line 54 of file AI.java.

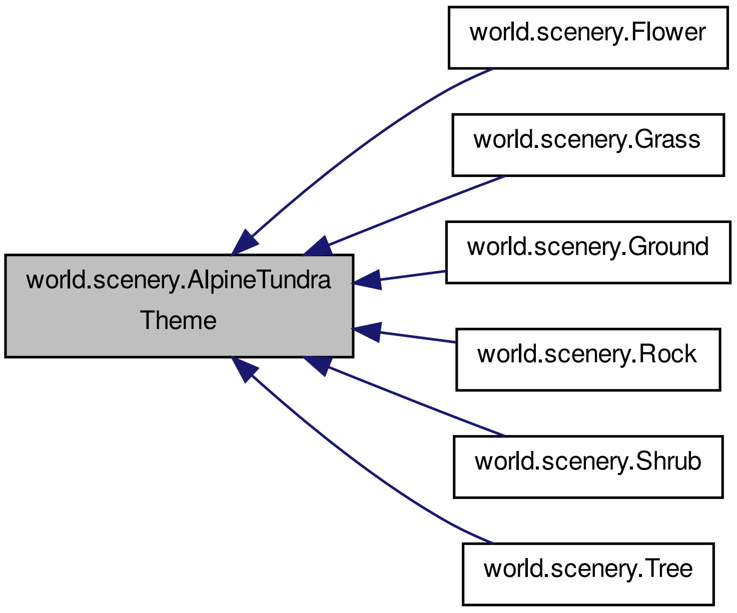
The documentation for this class was generated from the following file:

- gameplay/ai/AI.java

7.2 world.scenery.AlpineTundraTheme Interface Reference

Interface used to define that a Scenery object have an Alpine Tundra style.

Inheritance diagram for world.scenery.AlpineTundraTheme:



Static Public Attributes

- static String ALPINE_TUNDRA = "alpinetundra"

  String used to tell in which file is the sprite matching the Alpine Tundra style.

7.2.1 Detailed Description

Interface used to define that a Scenery object have an Alpine Tundra style.

Definition at line 7 of file AlpineTundraTheme.java.

7.2.2 Member Data Documentation

7.2.2.1 String world.scenery.AlpineTundraTheme.ALPINE_TUNDRA = "alpinetundra" `[static]`

String used to tell in which file is the sprite matching the Alpine Tundra style.

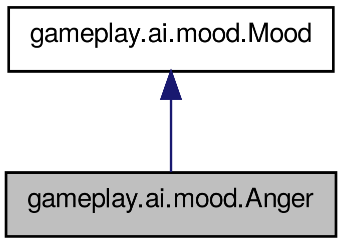
Definition at line 13 of file AlpineTundraTheme.java.

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

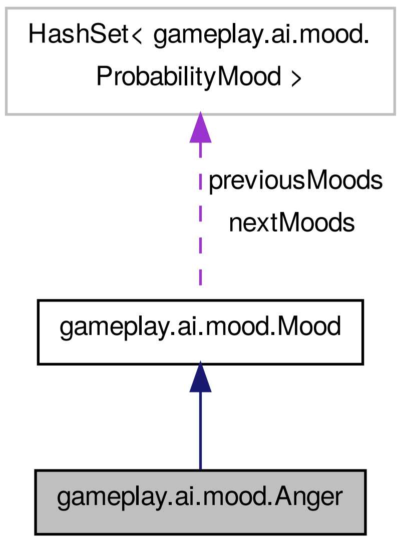
- world/scenery/AlpineTundraTheme.java

7.3 gameplay.ai.mood.Anger Class Reference

Inheritance diagram for gameplay.ai.mood.Anger:



Collaboration diagram for gameplay.ai.mood.Anger:



Public Member Functions

- Anger ()

Additional Inherited Members

7.3.1 Detailed Description

Definition at line 5 of file Anger.java.

7.3.2 Constructor & Destructor Documentation

7.3.2.1 gameplay.ai.mood.Anger.Anger ( )

Definition at line 7 of file Anger.java.

```
7                     {
8           super(10, 6);
9       }
```

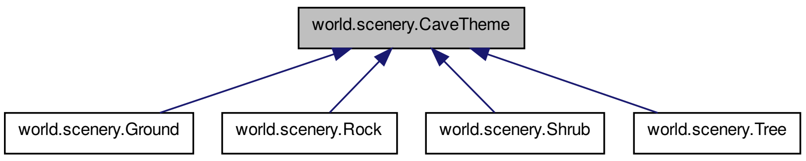
The documentation for this class was generated from the following file:

- gameplay/ai/mood/Anger.java

7.4 world.scenery.CaveTheme Interface Reference

Interface used to tell that a Scenery object has a Cave style.

Inheritance diagram for world.scenery.CaveTheme:



Static Public Attributes

- static String CAVE = "cave"

  String used to tell in which file is the sprite matching the Cave style.

7.4.1 Detailed Description

Interface used to tell that a Scenery object has a Cave style.

Definition at line 7 of file CaveTheme.java.

7.4.2 Member Data Documentation

7.4.2.1 String world.scenery.CaveTheme.CAVE = "cave" [static]

String used to tell in which file is the sprite matching the Cave style.

Definition at line 13 of file CaveTheme.java.

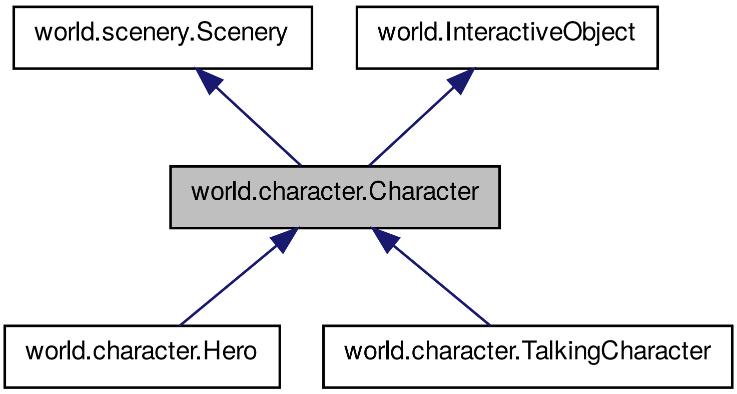
The documentation for this interface was generated from the following file:

- world/scenery/CaveTheme.java

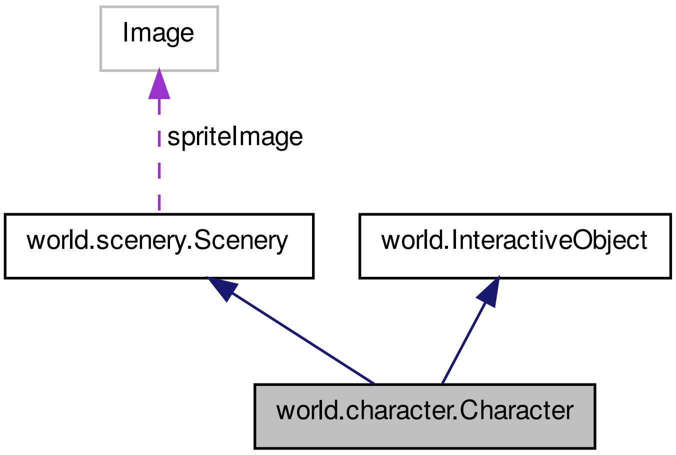
7.5 world.character.Character Class Reference

Class used to handle a character (player or non-player)

Inheritance diagram for world.character.Character:



Collaboration diagram for world.character.Character:



Public Member Functions

- Character (final String imagePath)

    Character class constructor.
- void jumpX (int distance)

    Teleport the character horizontally regarding his current position.

- void jumpY (int distance)

  Teleport the character vertically regarding his current position.
- void jump (int distanceX, int distanceY)

  Teleport the character regarding his current position.
- void moveX (int distance)

  Slowly horizontally move the character to a certain point regarding his current position.
- void moveY (int distance)

  Slowly vertically move the character to a certain point regarding his current position.
- void move (int distanceX, int distanceY)

  Slowly move the character to a certain point regarding his current position.
- void triggerAction ()

  Method to be called when the player is near the object and presses the action key.
- void playerNear ()

  Method to be called when the player is near the object.
- void playerEnter ()

  Method to be called when the player was near and is now over the object.
- void playerLeave ()

  Method to be called when the player was over and is now near the object.
- boolean isTraversable ()

  Returns true if the player is able to pass through the object and false if the player is not able to cross the object.

Protected Attributes

- int posX

  Horizontal position of the character.
- int posY

  Vertical position of the character.

Static Protected Attributes

- static final int CHARACTER_HEIGHT = 300

  Height of the character's image (fixed)
- static final int CHARACTER_WIDTH = 170

  Width of the character's image (fixed)

Additional Inherited Members

7.5.1 Detailed Description

Class used to handle a character (player or non-player)

A character can be a purely decorative character or a talking character or a player character (the hero)

Definition at line 13 of file Character.java.

7.5.2 Constructor & Destructor Documentation

7.5.2.1 world.character.Character.Character (final String *imagePath*)

Character class constructor.

Parameters

imagePath	Image filename representing the character

Definition at line 39 of file Character.java.

```
                                        {
        Philophobia.getVerbose().calls("Character class created", "world/characters/Character.java", "
    Character.Character(String)");

        super(image, CHARACTER_HEIGHT, CHARACTER_WIDTH);

    }
```

7.5.3 Member Function Documentation

7.5.3.1 boolean world.character.Character.isTraversable ( )

Returns true if the player is able to pass through the object and false if the player is not able to cross the object.

Implements world.InteractiveObject.

Definition at line 161 of file Character.java.

```
                                 {
        return false;
    }
```

7.5.3.2 void world.character.Character.jump (int *distanceX,* int *distanceY*)

Teleport the character regarding his current position.

Parameters

distanceX	Horizontal distance toward the character will be teleported
distanceY	Vertical distance toward the character will be teleported

Definition at line 73 of file Character.java.

```
                                                    {
        if(distanceX != 0 || distanceY != 0) {
            this.posX = this.posX + distance;
            this.posY = this.posY + distance;
            this.draw();
        }
    }
```

7.5.3.3 void world.character.Character.jumpX (int *distance*)

Teleport the character horizontally regarding his current position.

Parameters

distance	Horizontal distance toward the character will be teleported

Definition at line 50 of file Character.java.

```
                                      {
        if(distance != 0) {
            this.posX = this.posX + distance;
            this.draw();
        }
    }
```

7.5.3.4 void world.character.Character.jumpY (int *distance*)

Teleport the character vertically regarding his current position.

Parameters

distance	Vertical distance toward the character will be teleported

Definition at line 61 of file Character.java.

```
                                {
        if(distance != 0) {
            this.posY = this.posY + distance;
            this.draw();
        }
    }
```

7.5.3.5 void world.character.Character.move (int *distanceX*, int *distanceY*)

Slowly move the character to a certain point regarding his current position.

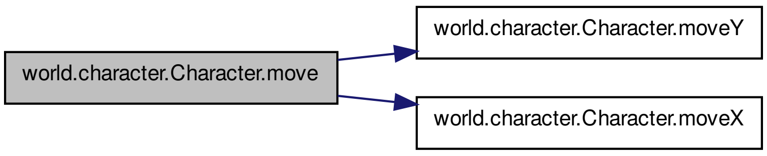
Parameters

distanceX	Horizontal distance toward the character will be moved
distanceY	Vertical distance toward the character will be moved

Definition at line 118 of file Character.java.

```
                                                    {

        if(distanceX != 0 || distanceY != 0) {
            int stepX = (distanceX < 0)? -1 : 1;
            int stepY = (distanceY < 0)? -1 : 1;

            this.jump(stepX, stepY);

            if(distanceX - stepX == 0) {
                moveY(distanceY - stepY);
                return;
            }

            if(distanceY - stepY == 0) {
                moveX(distanceX - stepX);
                return;
            }

            move(distanceX - stepX, distanceY - stepY);
        }

    }
```

Here is the call graph for this function:


world.character.Character.move
world.character.Character.moveY
world.character.Character.moveX

7.5.3.6 void world.character.Character.moveX (int *distance*)

Slowly horizontally move the character to a certain point regarding his current position.

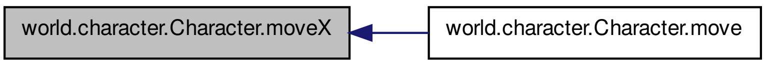
Parameters

distance	Horizontal distance toward the character will be moved

Definition at line 85 of file Character.java.

```
                                     {

        if(distance != 0) {
            int step = (distance < 0)? -1 : 1;

            this.jumpX(step);

            this.moveX(distance - step);
        }

    }
```

Here is the caller graph for this function:


world.character.Character.moveX
world.character.Character.move

7.5.3.7 void world.character.Character.moveY (int *distance*)

Slowly vertically move the character to a certain point regarding his current position.

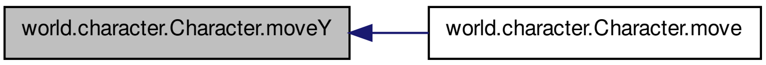
Parameters

distance	Vertical distance toward the character will be moved

Definition at line 101 of file Character.java.

```
                                     {

        if(distance != 0) {
            int step = (distance < 0)? -1 : 1;

            this.jumpY(step);

            this.moveY(distance - step);
        }

    }
```

Here is the caller graph for this function:


world.character.Character.moveY
world.character.Character.move

7.5.3.8 void world.character.Character.playerEnter ( )

Method to be called when the player was near and is now over the object.

Implements world.InteractiveObject.

Definition at line 153 of file Character.java.

```
153                                      {
154
155      }
```

7.5.3.9 void world.character.Character.playerLeave ( )

Method to be called when the player was over and is now near the object.

Implements world.InteractiveObject.

Definition at line 157 of file Character.java.

```
157                                      {
158
159      }
```

7.5.3.10 void world.character.Character.playerNear ( )

Method to be called when the player is near the object.

Implements world.InteractiveObject.

Definition at line 149 of file Character.java.

```
149                                     {
150
151      }
```

7.5.3.11 void world.character.Character.triggerAction ( )

Method to be called when the player is near the object and presses the action key.

Implements world.InteractiveObject.

Definition at line 145 of file Character.java.

```
145                                        {
146
147      }
```

7.5.4 Member Data Documentation

7.5.4.1 final int world.character.Character.CHARACTER_HEIGHT = 300 `[static]`, `[protected]`

Height of the character's image (fixed)

Definition at line 18 of file Character.java.

7.5.4.2 final int world.character.Character.CHARACTER_WIDTH = 170 `[static]`, `[protected]`

Width of the character's image (fixed)

Definition at line 23 of file Character.java.

7.5.4.3 int world.character.Character.posX `[protected]`

Horizontal position of the character.

Definition at line 28 of file Character.java.

7.5.4.4 int world.character.Character.posY `[protected]`

Vertical position of the character.

Definition at line 33 of file Character.java.

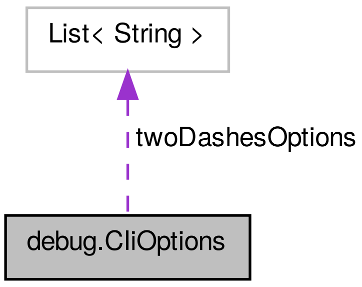
The documentation for this class was generated from the following file:

- world/character/Character.java

7.6 debug.CliOptions Class Reference

Class used to analyze the commands passed to the program.

Collaboration diagram for debug.CliOptions:



Public Member Functions

- CliOptions (String[] args)

  Constructor of the CliOption.
- int getVerboseLevel ()

  Return the verbose level passed as options to the program.
- boolean isHelpQueried ()

  Return true if the help has been queried or false if not.

Protected Member Functions

- void checkVerboseLevel ()

  Check the verbose level considering the options with one dash.
- void checkHelp ()

  Chech if the help mode has been queried considering the options with on or two dashes.

Protected Attributes

- int verboseLevel

  Level of verbosing detected due to the args passed to the program.
- boolean helpQueried

  Field equal to true if the help has been queried, false if not.
- String oneDashOptions

  One letter options passed to the program (corresponding to "one dash" options)
- List< String > twoDashesOptions

  List of two dashes options passed to the program.

7.6.1 Detailed Description

Class used to analyze the commands passed to the program.

For now, the verbose "-v" can be passed to the program (once or several times) and the help (containing the list of options available) can be queried via –help or -h

You can set a level of verbose mode from 0 to 5, you just have to put -v for verbose mode level 1, -vv for level 2 and so on

Definition at line 18 of file CliOptions.java.

7.6.2 Constructor & Destructor Documentation

7.6.2.1 debug.CliOptions.CliOptions (String[] *args*)

Constructor of the CliOption.

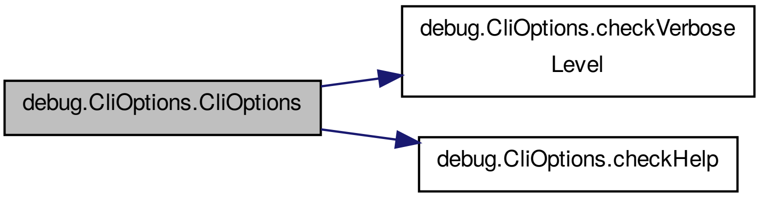
Parameters

args	Options passed to the program when launching it

Definition at line 52 of file CliOptions.java.

```
52                                     {
53 
54         oneDashOptions = new String();
55         twoDashesOptions = new LinkedList<String>();
56 
57         verboseLevel = 0;
58         helpQueried = false;
59 
60         // Check the arguments passed to the program
61         for(String str : args) {
62             // Check if an option is set
63             if(str.charAt(0) == '-' && str.length() > 1) {
64 
65                 // This program support the fact that an option can be prfixed with either '-' or '--'
66                 if(str.charAt(1) == '-' && str.length() > 2) {
67 
68                     twoDashesOptions.add(str.substring(2));
69 
70                 } else {
71 
72                     oneDashOptions += str.substring(1);
73 
74                 }
75 
76 
77             }
78         }
79 
80         checkVerboseLevel();
81         checkHelp();
82 
83     }
```

Here is the call graph for this function:


debug.CliOptions.CliOptions
debug.CliOptions.checkVerbose Level
debug.CliOptions.checkHelp

7.6.3 Member Function Documentation

7.6.3.1 void debug.CliOptions.checkHelp ( ) [protected]

Chech if the help mode has been queried considering the options with on or two dashes.

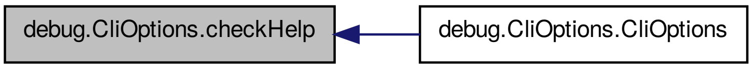
See Also

helpQueried

Definition at line 114 of file CliOptions.java.

```
                                {

        helpQueried = false;

        int oneDashOptionsLength = oneDashOptions.length();

        for(int i = 0 ; i < oneDashOptionsLength ; ++i) {
            if(oneDashOptions.charAt(i) == 'h') {
                helpQueried = true;
                return;
            }
        }

        ListIterator li = twoDashesOptions.listIterator();

        while(li.hasNext()) {
            if(li.next().equals("help")) {
                helpQueried = true;
                return;
            }
        }

    }
```

Here is the caller graph for this function:



7.6.3.2 void debug.CliOptions.checkVerboseLevel ( ) [protected]

Check the verbose level considering the options with one dash.

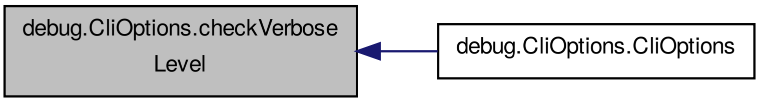
See Also

verboseLevel

Definition at line 89 of file CliOptions.java.

```
                                    {
        verboseLevel = 0;

        // Due to performance reasons, the length of the arguments are stored in a variable
        int oneDashOptionsLength = oneDashOptions.length();

        for(int i = 0 ; i < oneDashOptionsLength && verboseLevel < 5 ; ++i) {
            if(oneDashOptions.charAt(i) == 'v')
                ++verboseLevel;
        }

    }
```

Here is the caller graph for this function:


debug.CliOptions.checkVerbose Level
debug.CliOptions.CliOptions

7.6.3.3 int debug.CliOptions.getVerboseLevel ( )

Return the verbose level passed as options to the program.

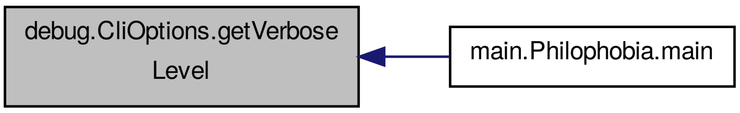
Returns

The verbose level

Definition at line 106 of file CliOptions.java.

```
                               {
         return verboseLevel;
    }
```

Here is the caller graph for this function:



7.6.3.4 boolean debug.CliOptions.isHelpQueried ( )

Return true if the help has been queried or false if not.

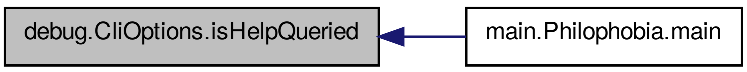
Returns

the helpQueried field

Definition at line 142 of file CliOptions.java.

```
                                    {
            return helpQueried;
        }
```

Here is the caller graph for this function:



7.6.4 Member Data Documentation

7.6.4.1 boolean debug.CliOptions.helpQueried [protected]

Field equal to true if the help has been queried, false if not.

If the help has been queried, then the game doesn't launch but instead print all the options that can be passed to the program

Definition at line 34 of file CliOptions.java.

7.6.4.2 String debug.CliOptions.oneDashOptions [protected]

One letter options passed to the program (corresponding to "one dash" options)

Definition at line 40 of file CliOptions.java.

7.6.4.3 List<String> debug.CliOptions.twoDashesOptions [protected]

List of two dashes options passed to the program.

Definition at line 45 of file CliOptions.java.

7.6.4.4 int debug.CliOptions.verboseLevel [protected]

Level of verbosing detected due to the args passed to the program.

See Also

Verbose
Verbose::verboseMode

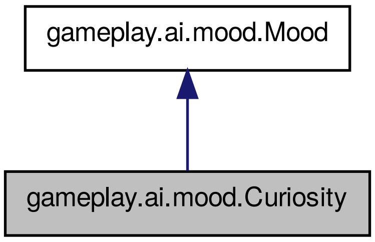
Definition at line 26 of file CliOptions.java.

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

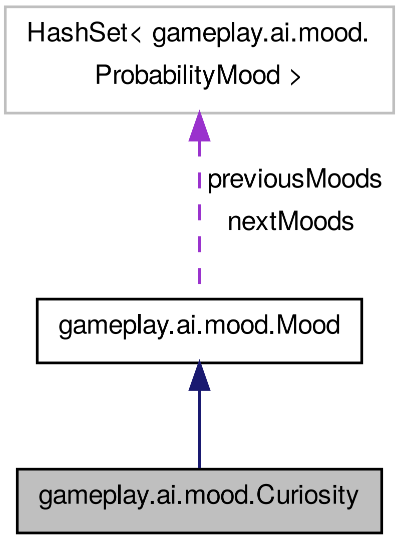
- debug/CliOptions.java

7.7 gameplay.ai.mood.Curiosity Class Reference

Inheritance diagram for gameplay.ai.mood.Curiosity:



Collaboration diagram for gameplay.ai.mood.Curiosity:


HashSet< gameplay.ai.mood. ProbabilityMood >
previousMoods
nextMoods
gameplay.ai.mood.Mood
gameplay.ai.mood.Curiosity

Public Member Functions

- Curiosity ()

Additional Inherited Members

7.7.1 Detailed Description

Definition at line 5 of file Curiosity.java.

7.7.2 Constructor & Destructor Documentation

7.7.2.1 gameplay.ai.mood.Curiosity.Curiosity ( )

Definition at line 7 of file Curiosity.java.

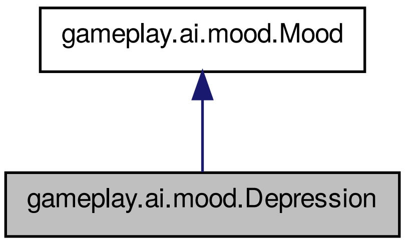
```
                        {
        super(10, 6);
    }
```

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

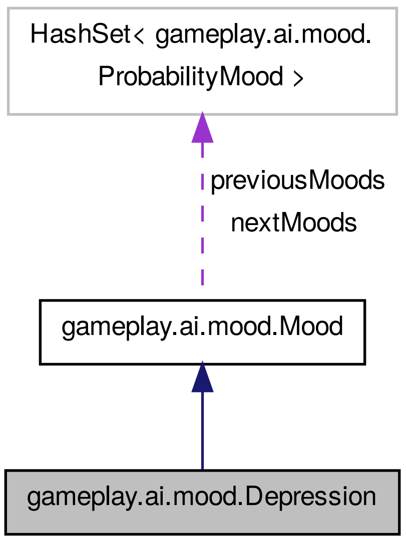
- gameplay/ai/mood/Curiosity.java

7.8 gameplay.ai.mood.Depression Class Reference

Inheritance diagram for gameplay.ai.mood.Depression:



Collaboration diagram for gameplay.ai.mood.Depression:


HashSet< gameplay.ai.mood.ProbabilityMood >
previousMoods
nextMoods
gameplay.ai.mood.Mood
gameplay.ai.mood.Depression

Public Member Functions

- Depression ()

Additional Inherited Members

7.8.1 Detailed Description

Definition at line 5 of file Depression.java.

7.8.2 Constructor & Destructor Documentation

7.8.2.1 gameplay.ai.mood.Depression.Depression ( )

Definition at line 7 of file Depression.java.

```
                        {
        super(10, 6);
    }
```

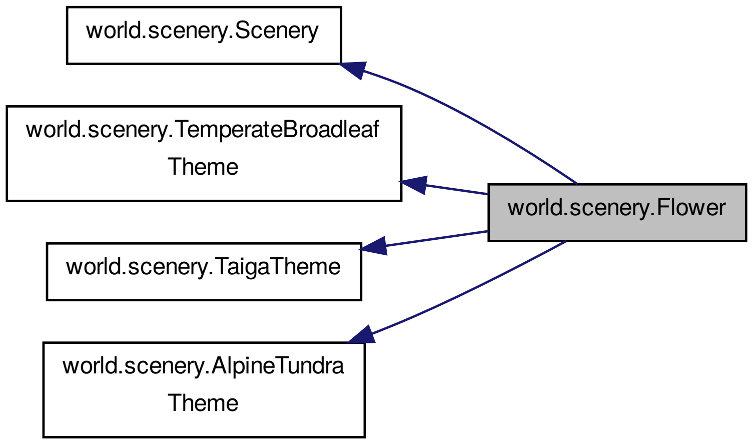
The documentation for this class was generated from the following file:

- gameplay/ai/mood/Depression.java

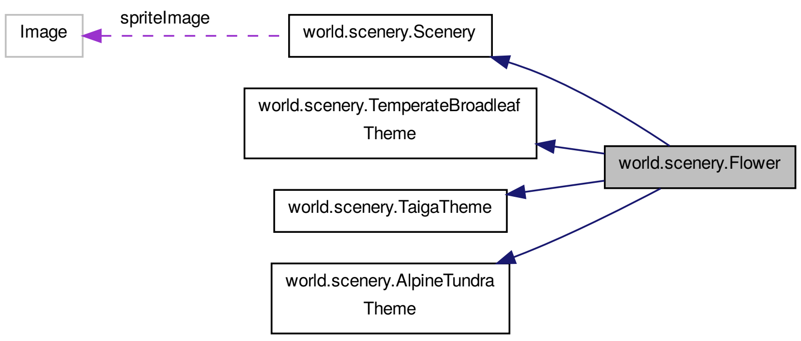
7.9 world.scenery.Flower Class Reference

Class used to handle a flower object.

Inheritance diagram for world.scenery.Flower:


world.scenery.Scenery
world.scenery.TemperateBroadleaf Theme
world.scenery.Flower
world.scenery.TaigaTheme
world.scenery.AlpineTundra Theme

Collaboration diagram for world.scenery.Flower:



Public Member Functions

- Flower (final String type)

    Flower class constructor.

Additional Inherited Members

7.9.1 Detailed Description

Class used to handle a flower object.

Definition at line 13 of file Flower.java.

7.9.2 Constructor & Destructor Documentation

7.9.2.1 world.scenery.Flower.Flower (final String *type*)

Flower class constructor.

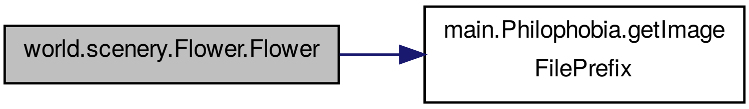
Parameters

type	Style of the flower

Definition at line 19 of file Flower.java.

```
                                    {

        super(Philophobia.getImageFilePrefix() + type + "flower.png");

    }
```

Here is the call graph for this function:


world.scenery.Flower.Flower
main.Philophobia.getImage FilePrefix

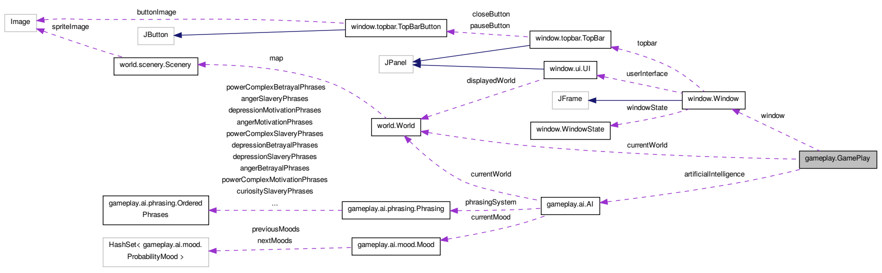
The documentation for this class was generated from the following file:

- world/scenery/Flower.java

7.10 gameplay.GamePlay Class Reference

Class handling the game play of this game.

Collaboration diagram for gameplay.GamePlay:



Public Member Functions

- GamePlay (CliOptions cliOptions)

  Constructor of the GamePlay class.

Protected Attributes

- World currentWorld

  World currently being played.
- String currentWorldName

  Name of the world currently being played.
- Window window

  The game window.
- AI artificialIntelligence

  Malicious artificial intelligence driving the player mad.

7.10.1 Detailed Description

Class handling the game play of this game.

It handle the creation of worlds, the movements of the main character and all the other characters, (including the constraints), the interactions between the main character and the other characters or the scenery

The GamePlay class also controls the user interface mode (loading screen, win, fail, etc. . .)

This class is the equivalent of the controller in the MVC paradigm.

See Also

world.World

Definition at line 25 of file GamePlay.java.

7.10.2 Constructor & Destructor Documentation

7.10.2.1 gameplay.GamePlay.GamePlay (**CliOptions** *cliOptions*)

Constructor of the GamePlay class.

Parameters

cliOptions	Opions for the game given via the command line

Definition at line 55 of file GamePlay.java.

```
55                                          {
56          Philophobia.getVerbose().information("Creating GamePlay class", "gameplay/Gameplay.java", "
     GamePlay.GamePlay()");
57
58          // The default world is the Temperate Broadleaf world
59          currentWorldName = "Temperate Broadleaf";
60          currentWorld = new World("temperatebroadleaf");
61
62          window = new Window();
63
64          try {
65              // Sleep during the length of the animation
66              Thread.sleep(2000);
67          } catch(InterruptedException e) {
68              Philophobia.getVerbose().warning("Sleep interrupted: " + e.getMessage(), "
     gameplay/GamePlay.java", "GamePlay.GamePlay(CliOptions)");
69          }
70
71          artificialIntelligence = new AI(currentWorld);
72
73          window.getUserInterface().setDisplayedWorld(currentWorld);
74          window.setGameState();
75      }
```

7.10.3 Member Data Documentation

7.10.3.1 AI gameplay.GamePlay.artificialIntelligence [protected]

Malicious artificial intelligence driving the player mad.

Definition at line 49 of file GamePlay.java.

7.10.3.2 World gameplay.GamePlay.currentWorld [protected]

World currently being played.

Definition at line 30 of file GamePlay.java.

7.10.3.3 String gameplay.GamePlay.currentWorldName [protected]

Name of the world currently being played.

It is used to detect in which world the player is and where are the other worlds

Definition at line 38 of file GamePlay.java.

7.10.3.4 Window gameplay.GamePlay.window [protected]

The game window.

Definition at line 43 of file GamePlay.java.

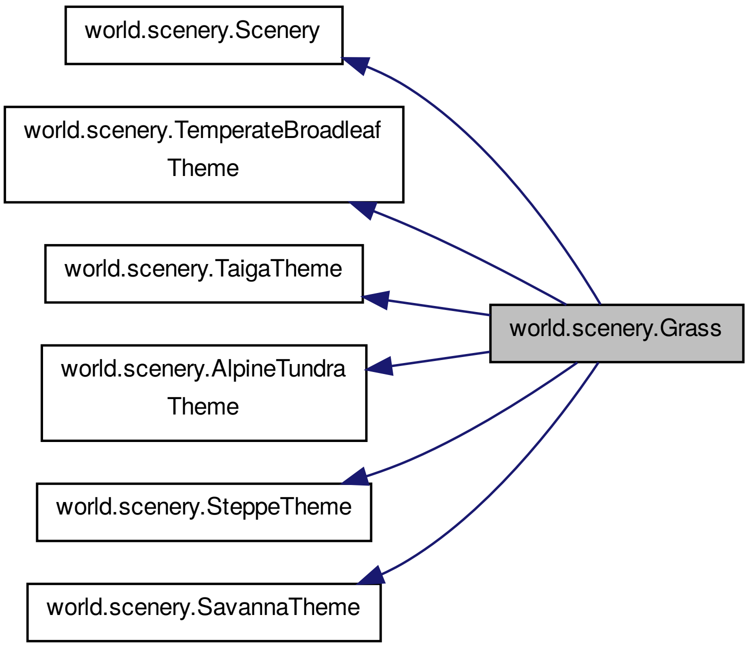
The documentation for this class was generated from the following file:

- gameplay/GamePlay.java

7.11 world.scenery.Grass Class Reference

Class used to handle a grass object.

Inheritance diagram for world.scenery.Grass:


world.scenery.Scenery
world.scenery.TemperateBroadleaf Theme
world.scenery.TaigaTheme
world.scenery.AlpineTundra Theme
world.scenery.SteppeTheme
world.scenery.SavannaTheme
world.scenery.Grass

Collaboration diagram for world.scenery.Grass:

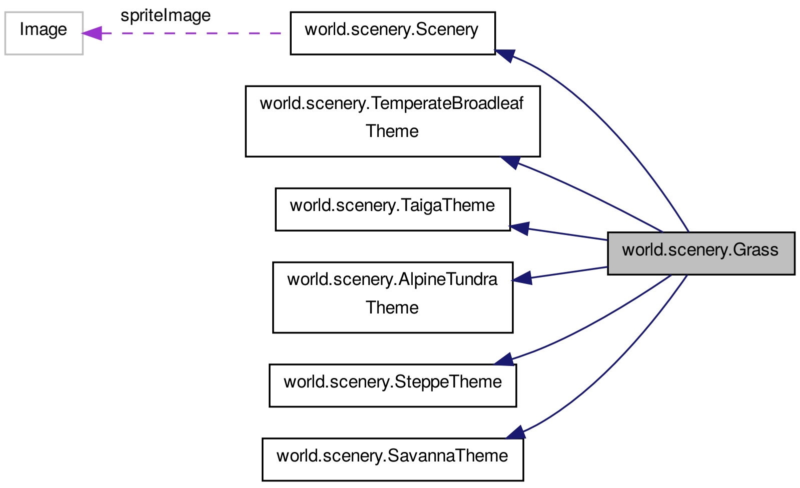


Image
spriteImage
world.scenery.Scenery
world.scenery.TemperateBroadleaf Theme
world.scenery.TaigaTheme
world.scenery.AlpineTundra Theme
world.scenery.SteppeTheme
world.scenery.SavannaTheme
world.scenery.Grass

Public Member Functions

- Grass (final String type)

  Grass class constructor.

Additional Inherited Members

7.11.1 Detailed Description

Class used to handle a grass object.

Definition at line 15 of file Grass.java.

7.11.2 Constructor & Destructor Documentation

7.11.2.1 world.scenery.Grass.Grass (final String *type*)

Grass class constructor.

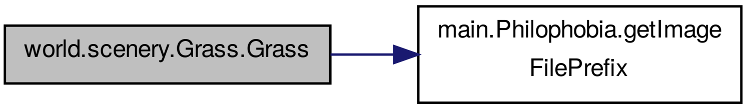
Parameters

type	Style of the grass

Definition at line 21 of file Grass.java.

```
                                    {

        super(Philophobia.getImageFilePrefix() + type + "grass.png");

    }
```

Here is the call graph for this function:


world.scenery.Grass.Grass
main.Philophobia.getImage FilePrefix

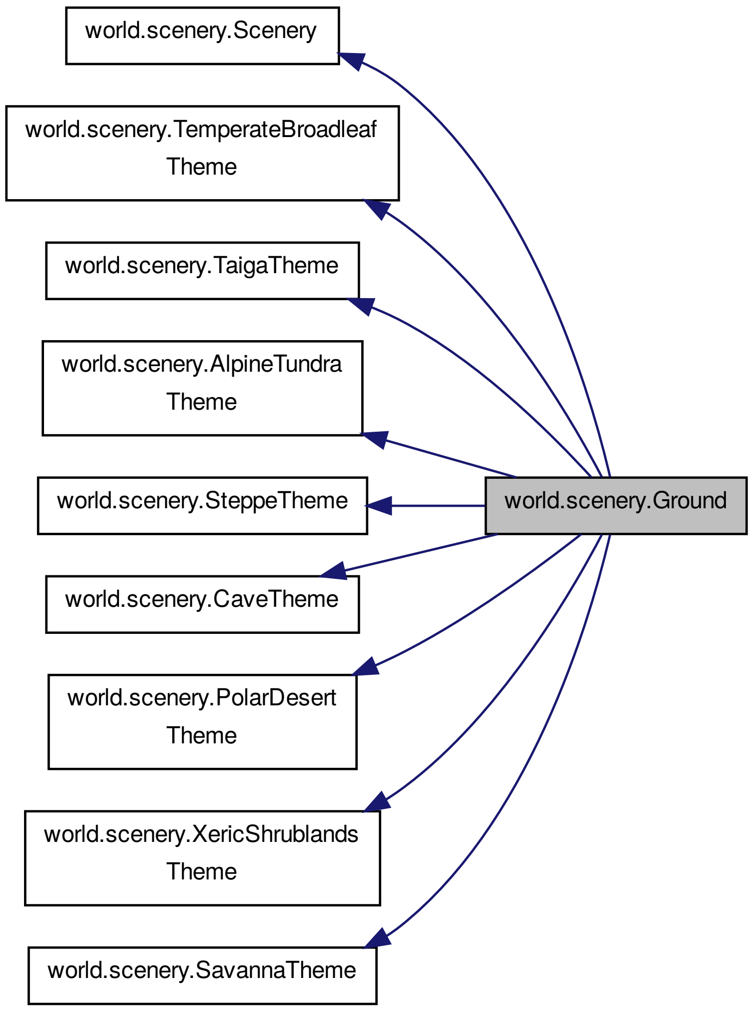
The documentation for this class was generated from the following file:

- world/scenery/Grass.java

7.12 world.scenery.Ground Class Reference

Class used to handle a ground object.

Inheritance diagram for world.scenery.Ground:


world.scenery.Scenery
world.scenery.TemperateBroadleaf Theme
world.scenery.TaigaTheme
world.scenery.AlpineTundra Theme
world.scenery.SteppeTheme
world.scenery.Ground
world.scenery.CaveTheme
world.scenery.PolarDesert Theme
world.scenery.XericShrublands Theme
world.scenery.SavannaTheme

Collaboration diagram for world.scenery.Ground:

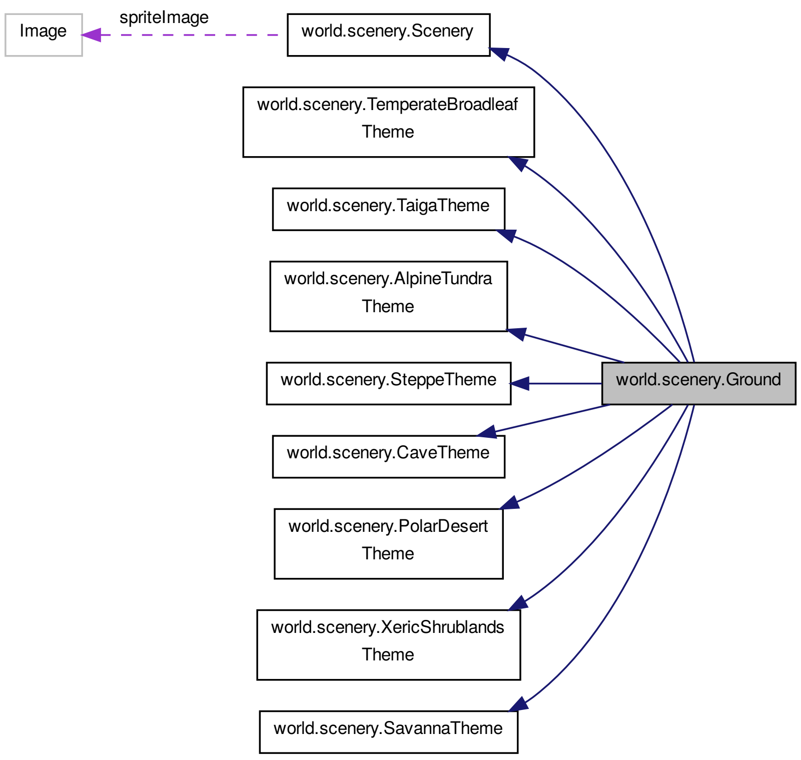


Image
spriteImage
world.scenery.Scenery
world.scenery.TemperateBroadleaf Theme
world.scenery.TaigaTheme
world.scenery.AlpineTundra Theme
world.scenery.SteppeTheme
world.scenery.Ground
world.scenery.CaveTheme
world.scenery.PolarDesert Theme
world.scenery.XericShrublands Theme
world.scenery.SavannaTheme

Public Member Functions

- Ground (final String type)

    Ground class constructor.

Additional Inherited Members

7.12.1 Detailed Description

Class used to handle a ground object.

Definition at line 18 of file Ground.java.

7.12.2 Constructor & Destructor Documentation

7.12.2.1 world.scenery.Ground.Ground (final String *type*)

Ground class constructor.

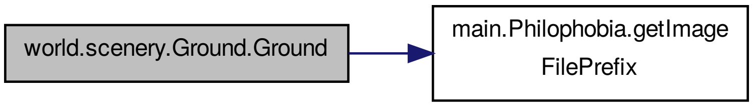
Parameters

type	Style of the ground

Definition at line 24 of file Ground.java.

```
                                    {

        super(Philophobia.getImageFilePrefix() + type + "ground.png");

    }
```

Here is the call graph for this function:

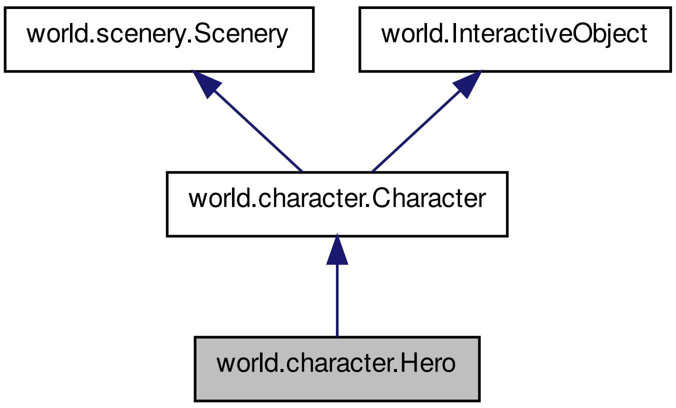

world.scenery.Ground.Ground
main.Philophobia.getImage FilePrefix

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

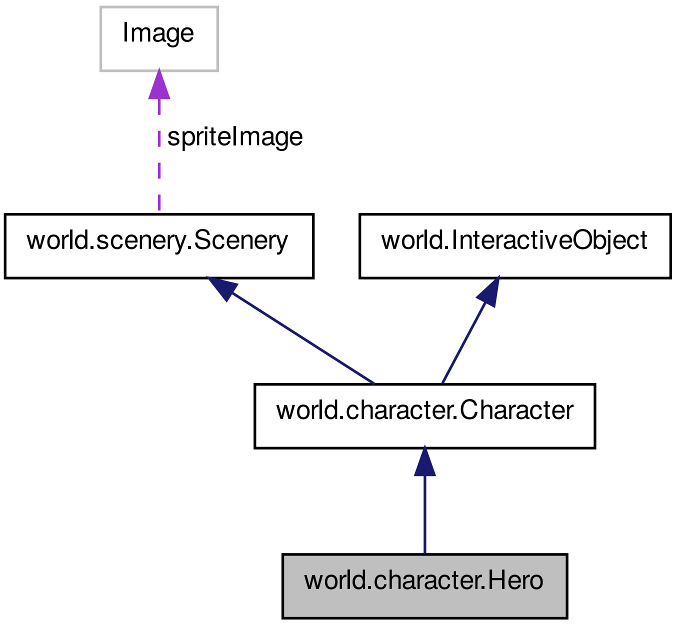
- world/scenery/Ground.java

7.13 world.character.Hero Class Reference

Inheritance diagram for world.character.Hero:



Collaboration diagram for world.character.Hero:



Public Member Functions

- Hero ()

Additional Inherited Members

7.13.1 Detailed Description

Definition at line 6 of file Hero.java.

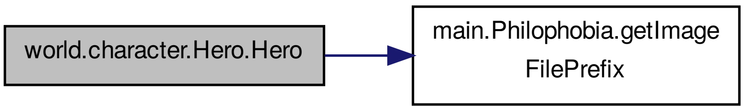
7.13.2 Constructor & Destructor Documentation

7.13.2.1 world.character.Hero.Hero ( )

Definition at line 8 of file Hero.java.

```
                {
        super(Philophobia.getImageFilePrefix() + "hero.png");
         Philophobia.getVerbose().information("Created Hero class", "world/character/Hero.java", "
      Hero.Hero()");
    }
```

Here is the call graph for this function:

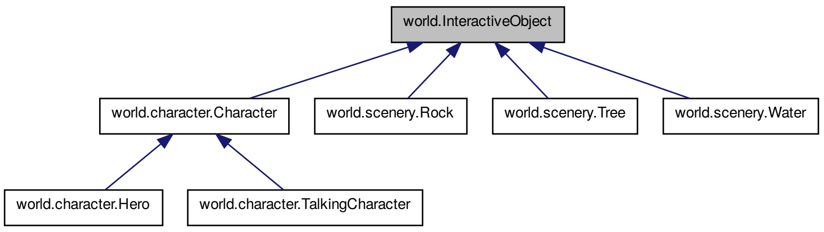

world.character.Hero.Hero
main.Philophobia.getImage FilePrefix

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

- world/character/Hero.java

7.14 world.InteractiveObject Interface Reference

Inheritance diagram for world.InteractiveObject:



Public Member Functions

- void triggerAction ()

  Method to be called when the player is near the object and presses the action key.
- void playerNear ()

  Method to be called when the player is near the object.
- void playerEnter ()

  Method to be called when the player was near and is now over the object.
- void playerLeave ()

  Method to be called when the player was over and is now near the object.
- boolean isTraversable ()

  Returns true if the player is able to pass through the object and false if the player is not able to cross the object.

7.14.1 Detailed Description

Definition at line 6 of file InteractiveObject.java.

7.14.2 Member Function Documentation

7.14.2.1 boolean world.InteractiveObject.isTraversable ( )

Returns true if the player is able to pass through the object and false if the player is not able to cross the object.

Implemented in world.character.Character, world.scenery.Tree, world.scenery.Rock, and world.scenery.Water.

7.14.2.2 void world.InteractiveObject.playerEnter ( )

Method to be called when the player was near and is now over the object.

Implemented in world.character.Character, world.scenery.Tree, world.scenery.Rock, and world.scenery.Water.

7.14.2.3 void world.InteractiveObject.playerLeave ( )

Method to be called when the player was over and is now near the object.

Implemented in world.character.Character, world.scenery.Tree, world.scenery.Rock, and world.scenery.Water.

7.14.2.4 void world.InteractiveObject.playerNear ( )

Method to be called when the player is near the object.

Implemented in world.character.Character, world.scenery.Tree, world.scenery.Rock, and world.scenery.Water.

7.14.2.5 void world.InteractiveObject.triggerAction ( )

Method to be called when the player is near the object and presses the action key.

Implemented in world.character.Character, world.scenery.Tree, world.scenery.Rock, world.character.Talking-Character, and world.scenery.Water.

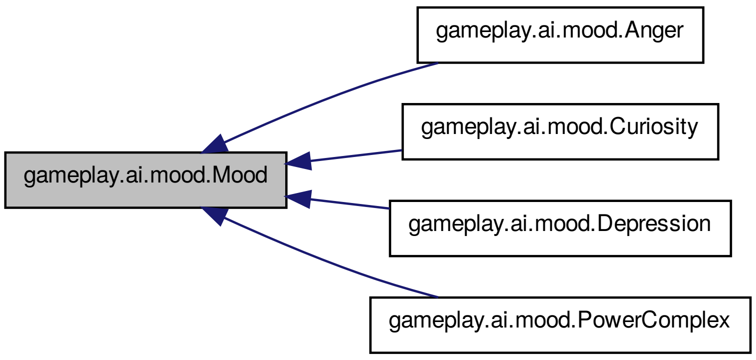
The documentation for this interface was generated from the following file:

- world/InteractiveObject.java

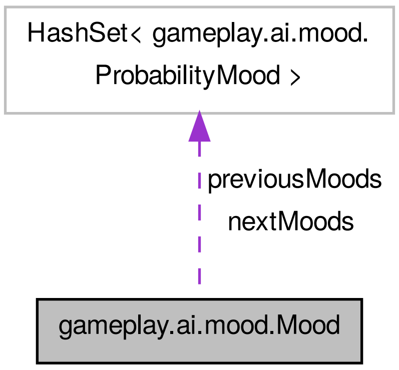
7.15 gameplay.ai.mood.Mood Class Reference

Abstract class representing an AI mood.

Inheritance diagram for gameplay.ai.mood.Mood:


gameplay.ai.mood.Anger
gameplay.ai.mood.Curiosity
gameplay.ai.mood.Mood
gameplay.ai.mood.Depression
gameplay.ai.mood.PowerComplex

Collaboration diagram for gameplay.ai.mood.Mood:



Public Member Functions

- Mood (int betrayalThreshold, int slaveryThreshold)

    Mood constructor.
- int getBetrayalThreshold ()

    Getter for the betrayal threshold field.
- int getSlaveryThreshold ()

    Getter for the slavery threshold field.
- int getMoodsBetrayalCount ()

    Getter for the betrayal count field.
- int getMoodsSlaveryThreshold ()

    Getter for the slavery count field.
- int incrementBetrayalCount ()

Increment and return the betrayal count field.
- int incrementSlaveryCount ()

  Increment and return the slavery count field.
- Class< Mood > getNextMood ()

  When the betrayal threshold is exceeded, this function is called and return a new mood given their probabilities.
- Class< Mood > getPreviousMood ()

  When the slavery threshold is exceeded, this function is called and return a new mood given their probabilities.

Protected Attributes

- int betrayalThreshold

  If the betrayal count goes above the betrayal threshold, then the robot must change its current mood (randomness required)
- int slaveryThreshold

  If the slavery count goes above the betrayal threshold, then the robot must change its current mood (randomness required)
- int moodsBetrayalCount

  Number of betrayals (requests from the AI not done by the user) done by the user when the AI was currently in this mood state.
- int moodsSlaveryCount

  Number of requests from the AI done positively by the user when the AI was in this mood state.
- HashSet< ProbabilityMood > nextMoods

  Associative array containing the possible moods if the number of betrayals (when the AI was in this mood state) is greater than its threshold with the probability associated.
- HashSet< ProbabilityMood > previousMoods

  Associative array containing the possible moods if the number of done actions asked by the AI is greater than its threshold with the probability associated.

7.15.1 Detailed Description

Abstract class representing an AI mood.

Definition at line 12 of file Mood.java.

7.15.2 Constructor & Destructor Documentation

7.15.2.1 gameplay.ai.mood.Mood.Mood (int *betrayalThreshold,* int *slaveryThreshold*)

Mood constructor.

Definition at line 66 of file Mood.java.

```
                                                                    {

        Philophobia.getVerbose().information("Creating new Mood class", "gameplay/ai/mood/Mood.java", "
    Mood.Mood(int, int)");

        this.betrayalThreshold = betrayalThreshold;
        this.slaveryThreshold = slaveryThreshold;

    }
```

7.15.3 Member Function Documentation

7.15.3.1 int gameplay.ai.mood.Mood.getBetrayalThreshold ( )

Getter for the betrayal threshold field.

See Also

betrayalThreshold

Definition at line 79 of file Mood.java.

```
                                {
        return betrayalThreshold;
    }
```

7.15.3.2 int gameplay.ai.mood.Mood.getMoodsBetrayalCount ( )

Getter for the betrayal count field.

See Also

moodsBetrayalCount

Definition at line 95 of file Mood.java.

```
                                 {
        return moodsBetrayalCount;
    }
```

7.15.3.3 int gameplay.ai.mood.Mood.getMoodsSlaveryThreshold ( )

Getter for the slavery count field.

See Also

moodsSlaveryCount

Definition at line 103 of file Mood.java.

```
                                     {
        return moodsSlaveryCount;
    }
```

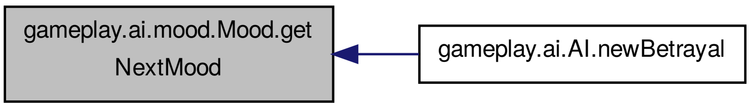
7.15.3.4 Class<**Mood**> gameplay.ai.mood.Mood.getNextMood ( )

When the betrayal threshold is exceeded, this function is called and return a new mood given their probabilities.

Definition at line 128 of file Mood.java.

```
                                {
        Iterator<ProbabilityMood> it = nextMoods.iterator();
        ProbabilityMood probMood = null;
        while(it.hasNext()) {

            probMood = it.next();
            if(Math.random() >= probMood.getProbability()) {
                return probMood.getMood();
            }

        }

        return probMood.getMood();
    }
```

Here is the caller graph for this function:


gameplay.ai.mood.Mood.get NextMood
gameplay.ai.AI.newBetrayal

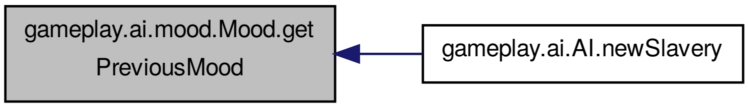
7.15.3.5 Class<**Mood**> gameplay.ai.mood.Mood.getPreviousMood ( )

When the slavery threshold is exceeded, this function is called and return a new mood given their probabilities.

Definition at line 148 of file Mood.java.

```
                                     {
        Iterator<ProbabilityMood> it = previousMoods.iterator();
        ProbabilityMood probMood = null;
        while(it.hasNext()) {

            probMood = it.next();
            if(Math.random() >= probMood.getProbability()) {
                return probMood.getMood();
            }

        }

        return probMood.getMood();
    }
```

Here is the caller graph for this function:


gameplay.ai.mood.Mood.get PreviousMood
gameplay.ai.AI.newSlavery

7.15.3.6 int gameplay.ai.mood.Mood.getSlaveryThreshold ( )

Getter for the slavery threshold field.

See Also

slaveryThreshold

Definition at line 87 of file Mood.java.

```
                                     {
        return slaveryThreshold;
    }
```

7.15.3.7 int gameplay.ai.mood.Mood.incrementBetrayalCount ( )

Increment and return the betrayal count field.

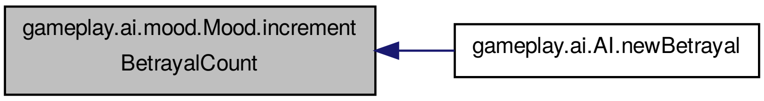
See Also

moodsBetrayalCount

Definition at line 111 of file Mood.java.

```
                                    {
        return ++moodsBetrayalCount;
    }
```

Here is the caller graph for this function:


gameplay.ai.mood.Mood.increment BetrayalCount
gameplay.ai.AI.newBetrayal

7.15.3.8 int gameplay.ai.mood.Mood.incrementSlaveryCount ( )

Increment and return the slavery count field.

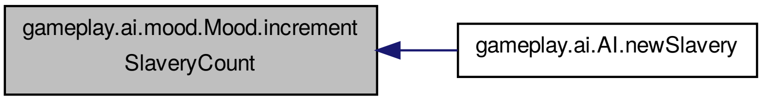
See Also

moodsSlaveryCount

Definition at line 119 of file Mood.java.

```
                                   {
        return ++moodsSlaveryCount;
    }
```

Here is the caller graph for this function:


gameplay.ai.mood.Mood.increment SlaveryCount
gameplay.ai.AI.newSlavery

7.15.4 Member Data Documentation

7.15.4.1 int gameplay.ai.mood.Mood.betrayalThreshold [protected]

If the betrayal count goes above the betrayal threshold, then the robot must change its current mood (randomness required)

Definition at line 20 of file Mood.java.

7.15.4.2 int gameplay.ai.mood.Mood.moodsBetrayalCount [protected]

Number of betrayals (requests from the AI not done by the user) done by the user when the AI was currently in this mood state.

Definition at line 37 of file Mood.java.

7.15.4.3 int gameplay.ai.mood.Mood.moodsSlaveryCount [protected]

Number of requests from the AI done positively by the user when the AI was in this mood state.

Definition at line 44 of file Mood.java.

7.15.4.4 HashSet<ProbabilityMood> gameplay.ai.mood.Mood.nextMoods [protected]

Associative array containing the possible moods if the number of betrayals (when the AI was in this mood state) is greater than its threshold with the probability associated.

Definition at line 53 of file Mood.java.

7.15.4.5 HashSet<ProbabilityMood> gameplay.ai.mood.Mood.previousMoods [protected]

Associative array containing the possible moods if the number of done actions asked by the AI is greater than its threshold with the probability associated.

Definition at line 61 of file Mood.java.

7.15.4.6 int gameplay.ai.mood.Mood.slaveryThreshold [protected]

If the slavery count goes above the betrayal threshold, then the robot must change its current mood (randomness required)

Definition at line 28 of file Mood.java.

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

- gameplay/ai/mood/Mood.java

7.16 gameplay.ai.phrasing.OrderedPhrases Class Reference

Class used to contain an ordered list of phrases.

Public Member Functions

- OrderedPhrases (final String filename)

    OrderedPhrases class constructor.

Protected Member Functions

- int getPhrasesNumber (final String filename)

    Method counting the number of line in filename.

Protected Attributes

- String[] phrases

  Array containing all the phrases corresponding to the current feeling and/or action.
- int[] phrasesOrder

  Order of the phrases array field.

7.16.1 Detailed Description

Class used to contain an ordered list of phrases.

It is ordered according to the last phrases that were spoken by the AI

The last phrases that were spoken by the AI are pushed at the bottom of the list

Definition at line 19 of file OrderedPhrases.java.

7.16.2 Constructor & Destructor Documentation

7.16.2.1 gameplay.ai.phrasing.OrderedPhrases.OrderedPhrases (final String *filename*)

OrderedPhrases class constructor.

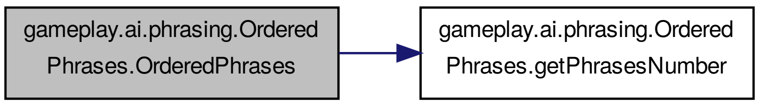
Parameters

filename	Path and name of the file contaning line by line the phrases

Definition at line 38 of file OrderedPhrases.java.

```
38                                              {
39          Philophobia.getVerbose().calls("Creating new OrderedPhrases class", "
        gameplay/ai/phrasing/OrderedPhrases.java", "OrderedPhrases.OrderedPhrases(String)");
40
41          int phrasesLength = getPhrasesNumber(filename);
42
43          phrases = new String[phrasesLength];
44          phrasesOrder = new int[phrasesLength];
45
46          try {
47              RandomAccessFile raFile = new RandomAccessFile(new File(filename), "r");
48
49              String currentLine = raFile.readLine();
50              for(int i = 0 ; currentLine != null ; ++i) {
51                  phrases[i] = currentLine;
52                  currentLine = raFile.readLine();
53              }
54          } catch(FileNotFoundException e) {
55              Philophobia.getVerbose().serious("File not found exception: " + e.getMessage(), "
        gameplay/ai/phrasing/OrderedPhrases.java", "OrderedPhrases.OrderedPhrases(String)");
56          } catch(IOException e) {
57              Philophobia.getVerbose().serious("IO exception: " + e.getMessage(), "
        gameplay/ai/phrasing/OrderedPhrases.java", "OrderedPhrases.OrderedPhrases(String)");
58          }
59      }
```

Here is the call graph for this function:


gameplay.ai.phrasing.Ordered Phrases.OrderedPhrases
gameplay.ai.phrasing.Ordered Phrases.getPhrasesNumber

7.16.3 Member Function Documentation

7.16.3.1 int gameplay.ai.phrasing.OrderedPhrases.getPhrasesNumber (final String *filename*) [protected]

Method counting the number of line in filename.

Parameters

filename	Path and name of the file

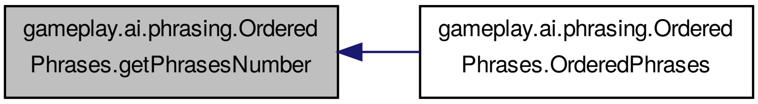
Returns

The number of phrases une the file or 0 if there is no phrases or an error

Definition at line 66 of file OrderedPhrases.java.

```
                                                          {

        try {
            // Opening filename i read mode
            RandomAccessFile raFile = new RandomAccessFile(new File(filename), "r");

            // Count the number of lines
            int i;
            for(i=0 ; raFile.readLine() != null ; ++i);
            return i;
        } catch(FileNotFoundException e) {
            Philophobia.getVerbose().serious("File not found exception: " + e.getMessage(), "
      gameplay/ai/phrasing/OrderedPhrases.java", "OrderedPhrases.OrderedPhrases(String)");
        } catch(IOException e) {
            Philophobia.getVerbose().serious("IO exception: " + e.getMessage(), "
      gameplay/ai/phrasing/OrderedPhrases.java", "OrderedPhrases.OrderedPhrases(String)");
        }
        return 0;
    }
```

Here is the caller graph for this function:


gameplay.ai.phrasing.Ordered Phrases.getPhrasesNumber
gameplay.ai.phrasing.Ordered Phrases.OrderedPhrases

7.16.4 Member Data Documentation

7.16.4.1 String [] gameplay.ai.phrasing.OrderedPhrases.phrases [protected]

Array containing all the phrases corresponding to the current feeling and/or action.

Definition at line 26 of file OrderedPhrases.java.

7.16.4.2 int [] gameplay.ai.phrasing.OrderedPhrases.phrasesOrder [protected]

Order of the phrases array field.

See Also

phrases

Definition at line 32 of file OrderedPhrases.java.

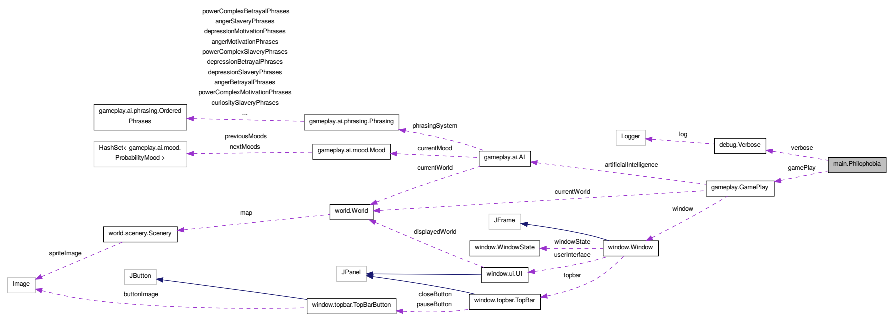
The documentation for this class was generated from the following file:

- gameplay/ai/phrasing/OrderedPhrases.java

7.17 main.Philophobia Class Reference

Main class.

Collaboration diagram for main.Philophobia:



Static Public Member Functions

- static void main (String[ ] args)

  Main function of the A3P program You can set options like verbose mode to the program.
- static Verbose getVerbose ()

  Getter for the verbose field.
- static String getImageFilePrefix ()

  Getter for the IMAGE_FILE_PREFIX field.
- static String getWorldFilePrefix ()

  Getter for the WORLD_FILE_PREFIX field.
- static String getPhrasesFilePrefix ()

Static Protected Attributes

- static String IMAGE_FILE_PREFIX = "src/images/"

  Directory containing all the image files.
- static String WORLD_FILE_PREFIX = "src/world/"

  Directory containing all the world files.
- static String PHRASES_FILE_PREFIX = "src/phrases/"

  Directory containing all the phrases files.
- static Verbose verbose

  Global variable used for debugging.
- static GamePlay gamePlay

  Main window of the program.

7.17.1 Detailed Description

Main class.

Definition at line 12 of file Philophobia.java.

7.17.2 Member Function Documentation

7.17.2.1 static String main.Philophobia.getImageFilePrefix ( ) [static]

Getter for the IMAGE_FILE_PREFIX field.

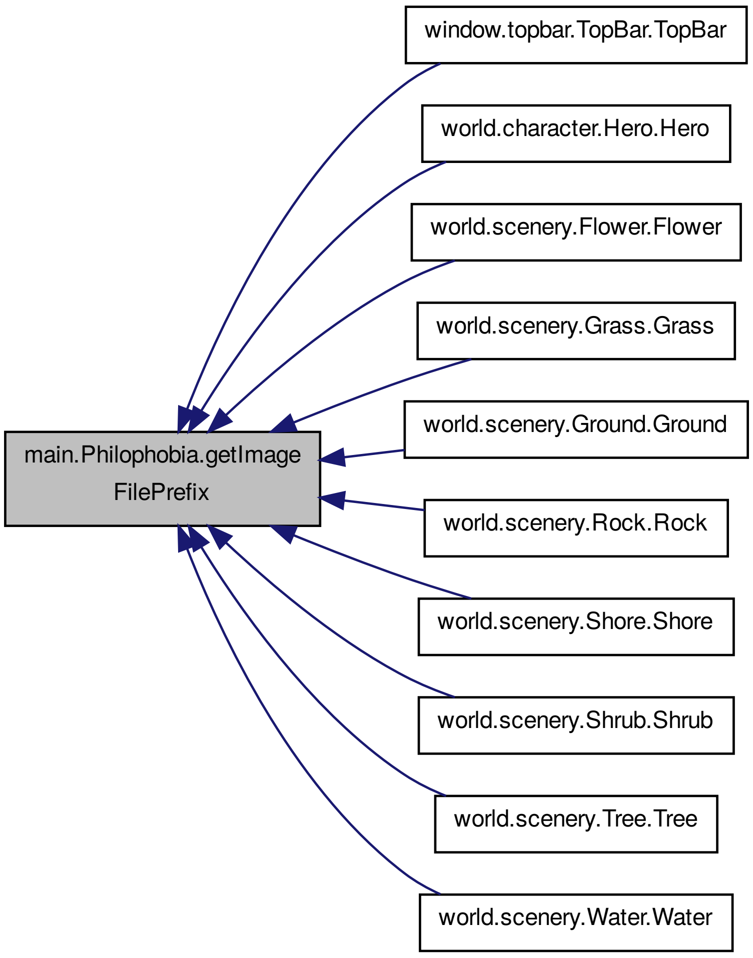
See Also

IMAGE_FILE_PREFIX

Definition at line 87 of file Philophobia.java.

```
                                          {
        return IMAGE_FILE_PREFIX;
    }
```

Here is the caller graph for this function:

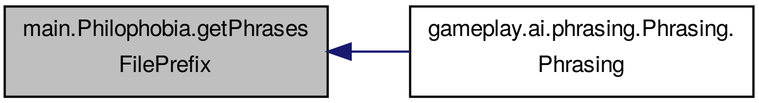

window.topbar.TopBar.TopBar
world.character.Hero.Hero
world.scenery.Flower.Flower
world.scenery.Grass.Grass
world.scenery.Ground.Ground
main.Philophobia.getImageFilePrefix
world.scenery.Rock.Rock
world.scenery.Shore.Shore
world.scenery.Shrub.Shrub
world.scenery.Tree.Tree
world.scenery.Water.Water

7.17.2.2 static String main.Philophobia.getPhrasesFilePrefix ( ) [static]

Definition at line 99 of file Philophobia.java.

```
                                                    {
            return PHRASES_FILE_PREFIX;
        }
```

Here is the caller graph for this function:


main.Philophobia.getPhrases FilePrefix
gameplay.ai.phrasing.Phrasing. Phrasing

7.17.2.3 static Verbose main.Philophobia.getVerbose ( ) [static]

Getter for the verbose field.

Returns

The main class verbose static property

Definition at line 79 of file Philophobia.java.

```
                                        {
            return verbose;
        }
```

7.17.2.4 static String main.Philophobia.getWorldFilePrefix ( ) [static]

Getter for the WORLD_FILE_PREFIX field.

See Also

WORLD_FILE_PREFIX

Definition at line 95 of file Philophobia.java.

```
                                                 {
            return WORLD_FILE_PREFIX;
        }
```

7.17.2.5 static void main.Philophobia.main (String[] *args*) [static]

Main function of the A3P program You can set options like verbose mode to the program.

See Also

debug.CliOptions

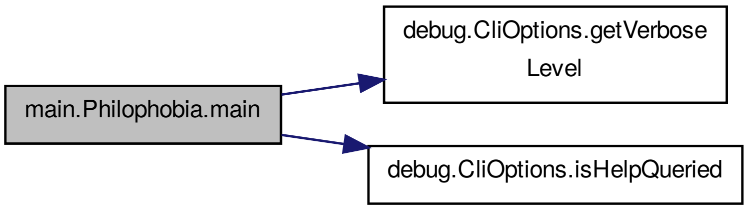
Parameters

args	Global program options (see above)

Definition at line 51 of file Philophobia.java.

```
                                            {

        CliOptions options = new CliOptions(args);

        Philophobia.verbose = new Verbose(options.getVerboseLevel());

        if(!options.isHelpQueried()) {
            // Philophobia.window = new Window();
            Philophobia.gamePlay = new GamePlay(options);
        } else {

            System.out.println("");
            System.out.println("Usage: philophobia [options]");
            System.out.println("");
            System.out.println("Options:");
            System.out.println("  Debugging options:");
            System.out.println("    -v                    Activate verbose mode (to be use multiple times
    for more verbosity)");
            System.out.println("    -h      --help        Print this help");
            System.out.println("");

        }

    }
```

Here is the call graph for this function:


main.Philophobia.main
debug.CliOptions.getVerbose Level
debug.CliOptions.isHelpQueried

7.17.3 Member Data Documentation

7.17.3.1 GamePlay main.Philophobia.gamePlay [static], [protected]

Main window of the program.

Game handler

Definition at line 42 of file Philophobia.java.

7.17.3.2 String main.Philophobia.IMAGE_FILE_PREFIX = "src/images/" [static], [protected]

Directory containing all the image files.

Definition at line 17 of file Philophobia.java.

7.17.3.3 String main.Philophobia.PHRASES_FILE_PREFIX = "src/phrases/" [static], [protected]

Directory containing all the phrases files.

Definition at line 27 of file Philophobia.java.

7.17.3.4 Verbose main.Philophobia.verbose `[static], [protected]`

Global variable used for debugging.

Definition at line 32 of file Philophobia.java.

7.17.3.5 String main.Philophobia.WORLD_FILE_PREFIX = "src/world/" `[static], [protected]`

Directory containing all the world files.

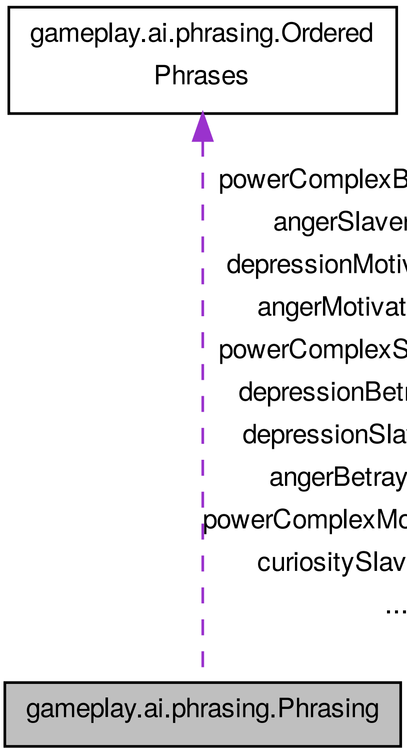
Definition at line 22 of file Philophobia.java.

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

- main/Philophobia.java

7.18 gameplay.ai.phrasing.Phrasing Class Reference

Collaboration diagram for gameplay.ai.phrasing.Phrasing:


gameplay.ai.phrasing.Ordered Phrases
powerComplexBetrayalPhrases
angerSlaveryPhrases
depressionMotivationPhrases
angerMotivationPhrases
powerComplexSlaveryPhrases
depressionBetrayalPhrases
depressionSlaveryPhrases
angerBetrayalPhrases
powerComplexMotivationPhrases
curiositySlaveryPhrases
...
gameplay.ai.phrasing.Phrasing

Public Member Functions

- Phrasing ()

  Phrasing class constructor.

Protected Attributes

- OrderedPhrases curiosityMotivationPhrases

  Attribute containing the AI phrases for the curiosity Feeling that aims to motivate the player to do some actions.
- OrderedPhrases curiosityBetrayalPhrases

  Attribute containing the AI phrases for the curiosity Feeling that are said when the player refused to do the asked action.
- OrderedPhrases curiositySlaveryPhrases

  Attribute containing the AI phrases for the curiosity Feeling that are said when the player did the asked action.
- OrderedPhrases angerMotivationPhrases

  Attribute containing the AI phrases for the anger Feeling that aims to motivate the player to do some actions.
- OrderedPhrases angerBetrayalPhrases

  Attribute containing the AI phrases for the anger Feeling that are said when the player refused to do the asked action.
- OrderedPhrases angerSlaveryPhrases

  Attribute containing the AI phrases for the anger Feeling that are said when the player did the asked action.
- OrderedPhrases depressionMotivationPhrases

  Attribute containing the AI phrases for the depression Feeling that aims to motivate the player to do some actions.
- OrderedPhrases depressionBetrayalPhrases

  Attribute containing the AI phrases for the depression Feeling that are said when the player refused to do the asked action.
- OrderedPhrases depressionSlaveryPhrases

  Attribute containing the AI phrases for the depression Feeling that are said when the player did the asked action.
- OrderedPhrases powerComplexMotivationPhrases

  Attribute containing the AI phrases for the power complex Feeling that aims to motivate the player to do some actions.
- OrderedPhrases powerComplexBetrayalPhrases

  Attribute containing the AI phrases for the power complex Feeling that are said when the player refused to do the asked action.
- OrderedPhrases powerComplexSlaveryPhrases

  Attribute containing the AI phrases for the power complex Feeling that are said when the player did the asked action.
- String actionAsked

  Field containing the asked action in a human readable format.

7.18.1 Detailed Description

Definition at line 6 of file Phrasing.java.

7.18.2 Constructor & Destructor Documentation

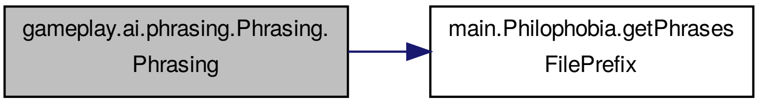
7.18.2.1 gameplay.ai.phrasing.Phrasing.Phrasing ( )

Phrasing class constructor.

Definition at line 101 of file Phrasing.java.

```
                      {
        Philophobia.getVerbose().calls("Creating Phrasing class", "gameplay/ai/phrasing/Phrasing.java", "
  Phrasing.Phrasing()");

        curiosityMotivationPhrases = new OrderedPhrases(Philophobia.
  getPhrasesFilePrefix() + "curiositymotivation.phrases");
        curiosityBetrayalPhrases = new OrderedPhrases(Philophobia.
  getPhrasesFilePrefix() + "curiositybetrayal.phrases");
        curiositySlaveryPhrases = new OrderedPhrases(Philophobia.
  getPhrasesFilePrefix() + "curiosityslavery.phrases");

        angerMotivationPhrases = new OrderedPhrases(Philophobia.getPhrasesFilePrefix(
  ) + "angermotivation.phrases");
        angerBetrayalPhrases = new OrderedPhrases(Philophobia.getPhrasesFilePrefix() +
  "angerbetrayal.phrases");
        angerSlaveryPhrases = new OrderedPhrases(Philophobia.getPhrasesFilePrefix() + "
  angerslavery.phrases");

        depressionMotivationPhrases = new OrderedPhrases(Philophobia.
  getPhrasesFilePrefix() + "depressionmotivation.phrases");
        depressionBetrayalPhrases = new OrderedPhrases(Philophobia.
  getPhrasesFilePrefix() + "depressionbetrayal.phrases");
        depressionSlaveryPhrases = new OrderedPhrases(Philophobia.
  getPhrasesFilePrefix() + "depressionslavery.phrases");

        powerComplexMotivationPhrases = new OrderedPhrases(Philophobia.
  getPhrasesFilePrefix() + "powercomplexmotivation.phrases");
        powerComplexBetrayalPhrases = new OrderedPhrases(Philophobia.
  getPhrasesFilePrefix() + "powercomplexbetrayal.phrases");
        powerComplexSlaveryPhrases = new OrderedPhrases(Philophobia.
  getPhrasesFilePrefix() + "powercomplexslavery.phrases");

    }
```

Here is the call graph for this function:



7.18.3 Member Data Documentation

7.18.3.1 String gameplay.ai.phrasing.Phrasing.actionAsked [protected]

Field containing the asked action in a human readable format.

Definition at line 96 of file Phrasing.java.

7.18.3.2 OrderedPhrases gameplay.ai.phrasing.Phrasing.angerBetrayalPhrases [protected]

Attribute containing the AI phrases for the anger Feeling that are said when the player refused to do the asked action.

Definition at line 41 of file Phrasing.java.

7.18.3.3 OrderedPhrases gameplay.ai.phrasing.Phrasing.angerMotivationPhrases [protected]

Attribute containing the AI phrases for the anger Feeling that aims to motivate the player to do some actions.

Definition at line 34 of file Phrasing.java.

7.18.3.4 OrderedPhrases gameplay.ai.phrasing.Phrasing.angerSlaveryPhrases [protected]

Attribute containing the AI phrases for the anger Feeling that are said when the player did the asked action.

Definition at line 48 of file Phrasing.java.

7.18.3.5 OrderedPhrases gameplay.ai.phrasing.Phrasing.curiosityBetrayalPhrases [protected]

Attribute containing the AI phrases for the curiosity Feeling that are said when the player refused to do the asked action.

Definition at line 20 of file Phrasing.java.

7.18.3.6 OrderedPhrases gameplay.ai.phrasing.Phrasing.curiosityMotivationPhrases [protected]

Attribute containing the AI phrases for the curiosity Feeling that aims to motivate the player to do some actions.

Definition at line 13 of file Phrasing.java.

7.18.3.7 OrderedPhrases gameplay.ai.phrasing.Phrasing.curiositySlaveryPhrases [protected]

Attribute containing the AI phrases for the curiosity Feeling that are said when the player did the asked action.

Definition at line 27 of file Phrasing.java.

7.18.3.8 OrderedPhrases gameplay.ai.phrasing.Phrasing.depressionBetrayalPhrases [protected]

Attribute containing the AI phrases for the depression Feeling that are said when the player refused to do the asked action.

Definition at line 62 of file Phrasing.java.

7.18.3.9 OrderedPhrases gameplay.ai.phrasing.Phrasing.depressionMotivationPhrases [protected]

Attribute containing the AI phrases for the depression Feeling that aims to motivate the player to do some actions.

Definition at line 55 of file Phrasing.java.

7.18.3.10 OrderedPhrases gameplay.ai.phrasing.Phrasing.depressionSlaveryPhrases [protected]

Attribute containing the AI phrases for the depression Feeling that are said when the player did the asked action.

Definition at line 69 of file Phrasing.java.

7.18.3.11 OrderedPhrases gameplay.ai.phrasing.Phrasing.powerComplexBetrayalPhrases [protected]

Attribute containing the AI phrases for the power complex Feeling that are said when the player refused to do the asked action.

Definition at line 83 of file Phrasing.java.

7.18.3.12 OrderedPhrases gameplay.ai.phrasing.Phrasing.powerComplexMotivationPhrases [protected]

Attribute containing the AI phrases for the power complex Feeling that aims to motivate the player to do some actions.

Definition at line 76 of file Phrasing.java.

7.18.3.13 OrderedPhrases gameplay.ai.phrasing.Phrasing.powerComplexSlaveryPhrases `[protected]`

Attribute containing the AI phrases for the power complex Feeling that are said when the player did the asked action.

Definition at line 90 of file Phrasing.java.

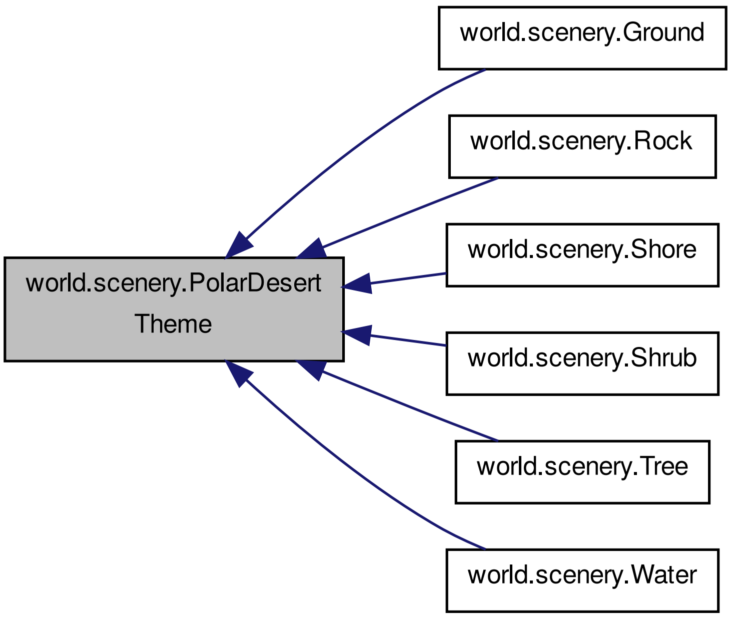
The documentation for this class was generated from the following file:

- gameplay/ai/phrasing/Phrasing.java

7.19 world.scenery.PolarDesertTheme Interface Reference

Interface used to define that a Scenery object has a Polar Desert style.

Inheritance diagram for world.scenery.PolarDesertTheme:



Static Public Attributes

- static String POLAR_DESERT = "polardesert"
  String used to tell in which file is the sprite matching the Polar Desert style.

7.19.1 Detailed Description

Interface used to define that a Scenery object has a Polar Desert style.

Definition at line 7 of file PolarDesertTheme.java.

7.19.2 Member Data Documentation

7.19.2.1 String world.scenery.PolarDesertTheme.POLAR_DESERT = "polardesert" `[static]`

String used to tell in which file is the sprite matching the Polar Desert style.

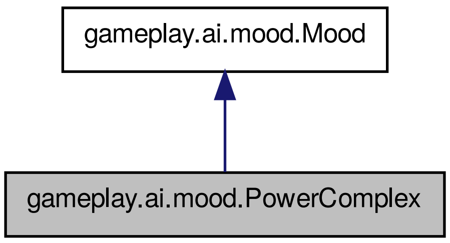
Definition at line 13 of file PolarDesertTheme.java.

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

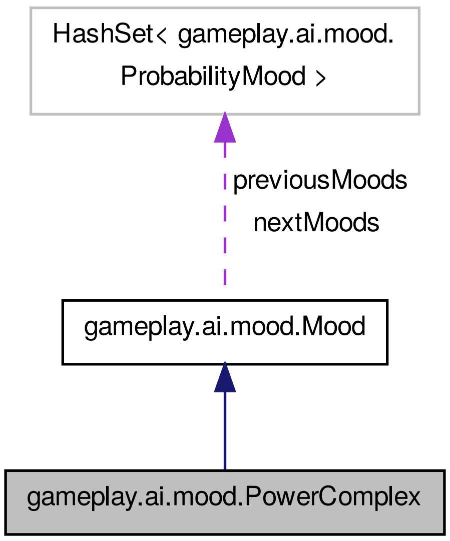
- world/scenery/PolarDesertTheme.java

7.20 gameplay.ai.mood.PowerComplex Class Reference

Inheritance diagram for gameplay.ai.mood.PowerComplex:



Collaboration diagram for gameplay.ai.mood.PowerComplex:


HashSet< gameplay.ai.mood.ProbabilityMood >
previousMoods
nextMoods
gameplay.ai.mood.Mood
gameplay.ai.mood.PowerComplex

Public Member Functions

- PowerComplex ()

Additional Inherited Members

7.20.1 Detailed Description

Definition at line 5 of file PowerComplex.java.

7.20.2 Constructor & Destructor Documentation

7.20.2.1 gameplay.ai.mood.PowerComplex.PowerComplex ( )

Definition at line 7 of file PowerComplex.java.

```
                        {
        super(10, 6);
    }
```

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

- gameplay/ai/mood/PowerComplex.java

7.21 gameplay.ai.mood.ProbabilityMood Class Reference

Class used to associate a mood with a probability.

Public Member Functions

- ProbabilityMood (Class< Mood > mood, double probability)

  Constructor of the probability mood class.
- Class< Mood > getMood ()

  Getter for the mood field.
- double getProbability ()

  Getter for the probability field.

Protected Attributes

- Class< Mood > mood

  Mood for which there is a probability to be chosen.
- double probability

  Probability of the mood to be chosen.

7.21.1 Detailed Description

Class used to associate a mood with a probability.

Definition at line 169 of file Mood.java.

7.21.2 Constructor & Destructor Documentation

7.21.2.1 gameplay.ai.mood.ProbabilityMood.ProbabilityMood (Class< **Mood** > *mood*, double *probability*)

Constructor of the probability mood class.

Definition at line 187 of file Mood.java.

```
                                                                 {
        this.mood = mood;
        this.probability = probability;
    }
```

7.21.3 Member Function Documentation

7.21.3.1 Class<**Mood**> gameplay.ai.mood.ProbabilityMood.getMood ( )

Getter for the mood field.

See Also

mood

Definition at line 196 of file Mood.java.

```
                                {
        return mood;
    }
```

7.21.3.2 double gameplay.ai.mood.ProbabilityMood.getProbability ( )

Getter for the probability field.

See Also

probability

Definition at line 204 of file Mood.java.

```
                                  {
        return probability;
    }
```

7.21.4 Member Data Documentation

7.21.4.1 Class<**Mood**> gameplay.ai.mood.ProbabilityMood.mood [protected]

Mood for which there is a probability to be chosen.

Definition at line 175 of file Mood.java.

7.21.4.2 double gameplay.ai.mood.ProbabilityMood.probability [protected]

Probability of the mood to be chosen.

Definition at line 181 of file Mood.java.

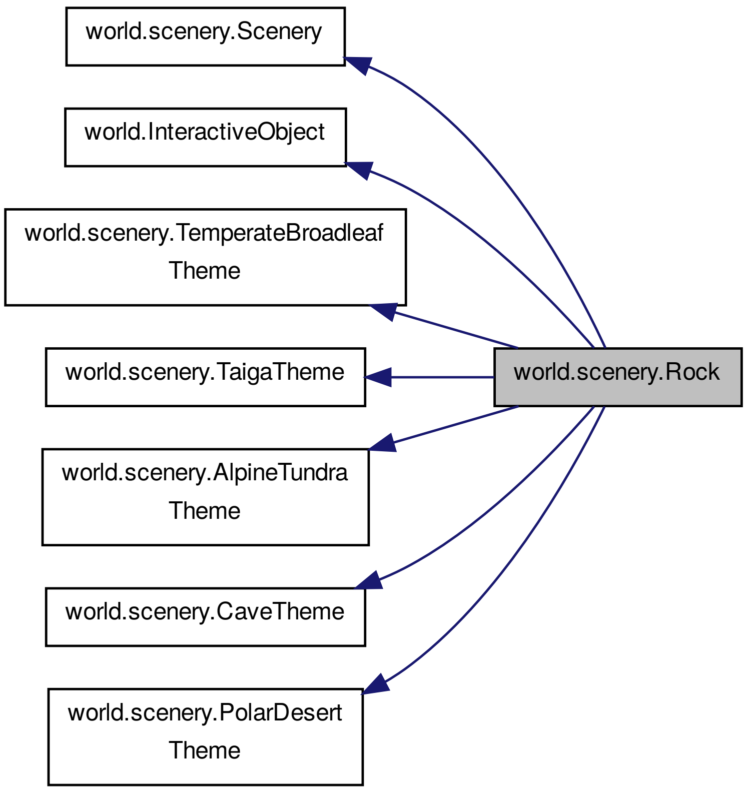
The documentation for this class was generated from the following file:

- gameplay/ai/mood/Mood.java

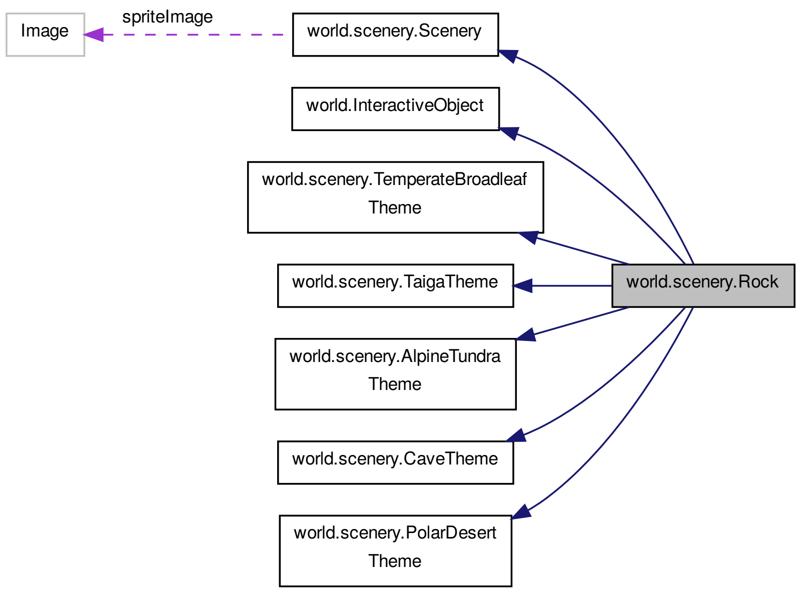
7.22 world.scenery.Rock Class Reference

Class used to handle a rock object.

Inheritance diagram for world.scenery.Rock:



Collaboration diagram for world.scenery.Rock:



Public Member Functions

- Rock (final String type)

  Rock class constructor.
- void triggerAction ()

  Method to be called when the player is near the object and presses the action key.
- void playerNear ()

  Method to be called when the player is near the object.
- void playerEnter ()

  Method to be called when the player was near and is now over the object.
- void playerLeave ()

  Method to be called when the player was over and is now near the object.
- boolean isTraversable ()

  Returns true if the player is able to pass through the object and false if the player is not able to cross the object.

Additional Inherited Members

7.22.1 Detailed Description

Class used to handle a rock object.

Definition at line 16 of file Rock.java.

7.22.2 Constructor & Destructor Documentation

7.22.2.1 world.scenery.Rock.Rock (final String *type*)

Rock class constructor.

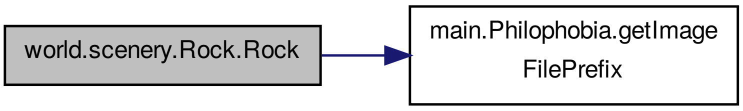
Parameters

type	Style of the rock

Definition at line 22 of file Rock.java.

```
                                {

        super(Philophobia.getImageFilePrefix() + type + "rock.png");

    }
```

Here is the call graph for this function:



7.22.3 Member Function Documentation

7.22.3.1 boolean world.scenery.Rock.isTraversable ( )

Returns true if the player is able to pass through the object and false if the player is not able to cross the object.

Implements world.InteractiveObject.

Definition at line 48 of file Rock.java.

```
                                {
        return false;
    }
```

7.22.3.2 void world.scenery.Rock.playerEnter ( )

Method to be called when the player was near and is now over the object.

Implements world.InteractiveObject.

Definition at line 40 of file Rock.java.

```
                            {

    }
```

7.22.3.3 void world.scenery.Rock.playerLeave ( )

Method to be called when the player was over and is now near the object.

Implements world.InteractiveObject.

Definition at line 44 of file Rock.java.

```
                            {

    }
```

7.22.3.4 void world.scenery.Rock.playerNear ( )

Method to be called when the player is near the object.

Implements world.InteractiveObject.

Definition at line 36 of file Rock.java.

```
36                                          {
37
38          }
```

7.22.3.5 void world.scenery.Rock.triggerAction ( )

Method to be called when the player is near the object and presses the action key.

Implements world.InteractiveObject.

Definition at line 32 of file Rock.java.

```
32                                              {
33
34          }
```

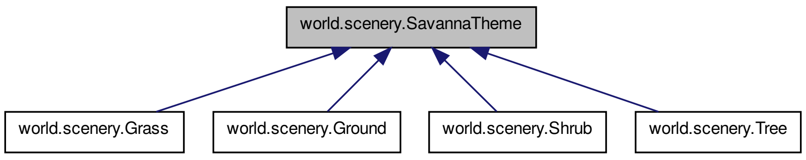
The documentation for this class was generated from the following file:

- world/scenery/Rock.java

7.23 world.scenery.SavannaTheme Interface Reference

Interface used to define that a Scenery object have a Savanna style.

Inheritance diagram for world.scenery.SavannaTheme:



Static Public Attributes

- static String SAVANNA = "savanna"

    String used to tell in which file is the sprite matching the Savanna style.

7.23.1 Detailed Description

Interface used to define that a Scenery object have a Savanna style.

Definition at line 7 of file SavannaTheme.java.

7.23.2 Member Data Documentation

7.23.2.1 String world.scenery.SavannaTheme.SAVANNA = "savanna" [static]

String used to tell in which file is the sprite matching the Savanna style.

Definition at line 13 of file SavannaTheme.java.

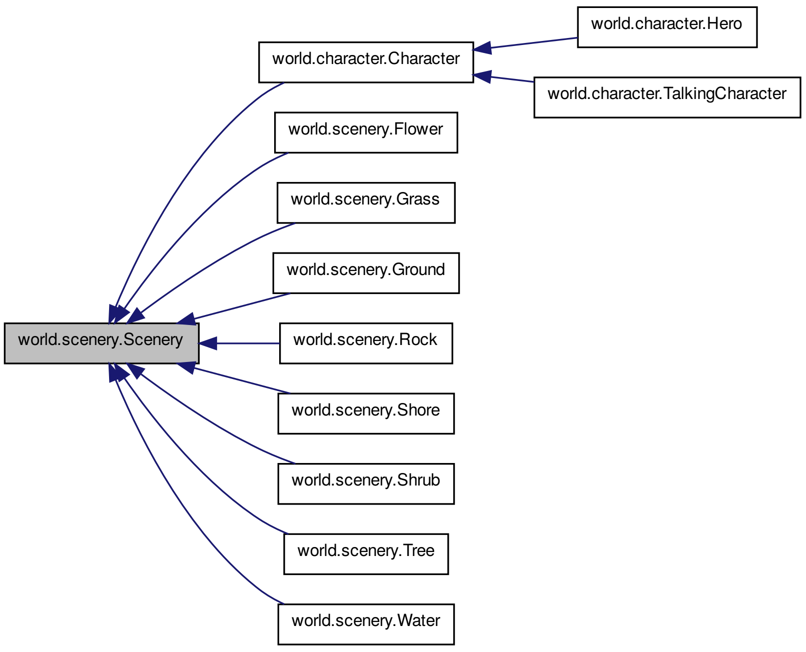
The documentation for this interface was generated from the following file:

- world/scenery/SavannaTheme.java

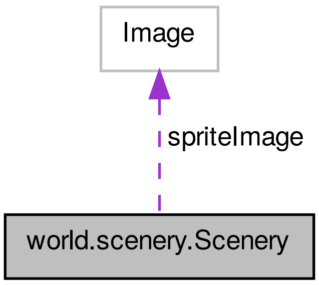
7.24 world.scenery.Scenery Class Reference

Class used to handle any world object.

Inheritance diagram for world.scenery.Scenery:



Collaboration diagram for world.scenery.Scenery:



Public Member Functions

- Scenery (final String spritePath, final int height, final int width)

  Scenery class constructor with height and width as parameters.
- Scenery (final String spritePath)

  Scenery class constructor with no optional parameters.
- Scenery (final String spritePath, final int height, final int width, final int xShift, final int yShift)

  Scenery class constructor with height, width, x shifting and y shifting as parameters.
- void drawScenery (Graphics g, final int xLocation, final int yLocation, ImageObserver obs)

Static Public Member Functions

- static int getSceneryHeight ()

  Getter for the SCENERY_HEIGHT static field.
- static int getSceneryWidth ()

  Getter for the SCENERY_WIDTH static field.

Protected Attributes

- Image spriteImage

  Image graphically representing the object.
- int height

  Image height.
- int width

  Image width.
- int xShift

  Horizontal shifting for the image.
- int yShift

  Vertical shifting for the image.
- boolean visible

  Boolean equals to true if the scenery is visible, false if the scenery is not.

Static Protected Attributes

- static int SCENERY_HEIGHT = 48

  Height of the graphical representation of a Scenery object.
- static int SCENERY_WIDTH = 48

  Width of the graphical representation of a Scenery object.

7.24.1 Detailed Description

Class used to handle any world object.

The Scenery class is used to handle the objects that are displayed in the graphical game world.

Definition at line 19 of file Scenery.java.

7.24.2 Constructor & Destructor Documentation

7.24.2.1 world.scenery.Scenery.Scenery (final String *spritePath*, final int *height*, final int *width*)

Scenery class constructor with height and width as parameters.

Parameters

spritePath	Path of the image used to represent the object
height	height of the object's pictural representaion
width	width of the object's pictural representation

Definition at line 68 of file Scenery.java.

```
                                                                    {
        Philophobia.getVerbose().calls("Creating Scenery object", "world/Scenery.java", "
    Scenery.Scenery(String, int, int)");

        try {
            spriteImage = ImageIO.read(new File(spritePath));
        } catch(IOException e) {
            Philophobia.getVerbose().warning("Scenery image \"" + spritePath + "\" load failed: " + e.
getMessage(), "world/Scenery.java", "Scenery.Scenery(String, int, int)");
        }

        this.height = height;
        this.width = width;

        xShift = 0;
        yShift = 0;
    }
```

7.24.2.2 world.scenery.Scenery.Scenery (final String *spritePath*)

Scenery class constructor with no optional parameters.

Parameters

spritePath	Path of the image used to represent the object

Definition at line 89 of file Scenery.java.

```
                                        {
        Philophobia.getVerbose().calls("Creating Scenery object", "world/Scenery.java", "
    Scenery.Scenery(String)");

        try {
            spriteImage = ImageIO.read(new File(spritePath));

            height = spriteImage.getHeight(null);
            width = spriteImage.getWidth(null);

        } catch(IOException e) {
                Philophobia.getVerbose().warning("Scenery image \"" + spritePath + "\"  load failed: " + e.
      getMessage(), "world/Scenery.java", "Scenery.Scenery(String)");
                height = 0;
                width = 0;
        }

        xShift = 0;
        yShift = 0;
    }
```

7.24.2.3 world.scenery.Scenery.Scenery (final String *spritePath,* final int *height,* final int *width,* final int *xShift,* final int *yShift*)

Scenery class constructor with height, width, x shifting and y shifting as parameters.

Parameters

spritePath	Path of the image used to represent the object
height	height of the object's pictural representation
width	width of the object's pictural representation
xShift	Horizontal shift of the image
yShift	Vertical shift or the image

Definition at line 117 of file Scenery.java.

```
          {
        Philophobia.getVerbose().calls("Creating Scenery object", "world/Scenery.java", "
      Scenery.Scenery(String, int, int, int, int)");

        try {
            spriteImage = ImageIO.read(new File(spritePath));
        } catch(IOException e) {
            Philophobia.getVerbose().warning("Scenery image load failed: " + e.getMessage(), "
      world/Scenery.java", "Scenery.Scenery(String, int, int, int, int)");
        }

        this.height = height;
        this.width = width;

        this.xShift = xShift;
        this.yShift = yShift;
    }
```

7.24.3 Member Function Documentation

7.24.3.1 void world.scenery.Scenery.drawScenery (Graphics *g,* final int *xLocation,* final int *yLocation,* ImageObserver *obs*)

Definition at line 133 of file Scenery.java.

```
                                                                                          {
        // drawImage(Image img, int x, int y, int width, int height, Observer obs);
        g.drawImage(spriteImage, xLocation + xShift, yLocation +
      yShift, width, height, obs);

        visible = true;
    }
```

7.24.3.2 static int world.scenery.Scenery.getSceneryHeight ( ) [static]

Getter for the SCENERY_HEIGHT static field.

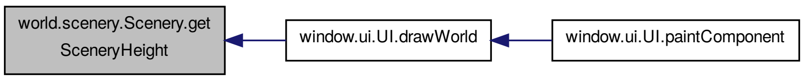
See Also

SCENERY_HEIGHT

Definition at line 144 of file Scenery.java.

```
                                      {
        return SCENERY_HEIGHT;
    }
```

Here is the caller graph for this function:


world.scenery.Scenery.getSceneryHeight
window.ui.UI.drawWorld
window.ui.UI.paintComponent

7.24.3.3 static int world.scenery.Scenery.getSceneryWidth ( ) [static]

Getter for the SCENERY_WIDTH static field.

See Also

SCENERY_WIDTH

Definition at line 152 of file Scenery.java.

```
                                     {
        return SCENERY_WIDTH;
    }
```

7.24.4 Member Data Documentation

7.24.4.1 int world.scenery.Scenery.height [protected]

Image height.

Definition at line 39 of file Scenery.java.

7.24.4.2 int world.scenery.Scenery.SCENERY_HEIGHT = 48 [static], [protected]

Height of the graphical representation of a Scenery object.

Definition at line 24 of file Scenery.java.

7.24.4.3 int world.scenery.Scenery.SCENERY_WIDTH = 48 [static], [protected]

Width of the graphical representation of a Scenery object.

Definition at line 29 of file Scenery.java.

7.24.4.4 Image world.scenery.Scenery.spriteImage [protected]

Image graphically representing the object.

Definition at line 34 of file Scenery.java.

7.24.4.5 boolean world.scenery.Scenery.visible `[protected]`

Boolean equals to true if the scenery is visible, false if the scenery is not.

Definition at line 59 of file Scenery.java.

7.24.4.6 int world.scenery.Scenery.width `[protected]`

Image width.

Definition at line 44 of file Scenery.java.

7.24.4.7 int world.scenery.Scenery.xShift `[protected]`

Horizontal shifting for the image.

Definition at line 49 of file Scenery.java.

7.24.4.8 int world.scenery.Scenery.yShift `[protected]`

Vertical shifting for the image.

Definition at line 54 of file Scenery.java.

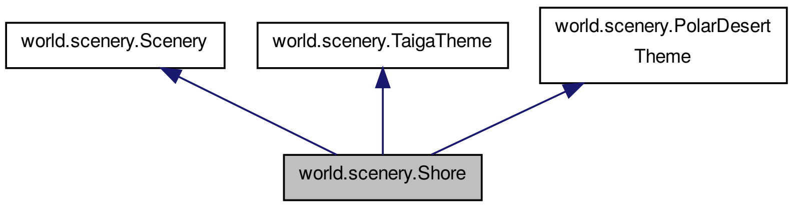
The documentation for this class was generated from the following file:

- world/scenery/Scenery.java

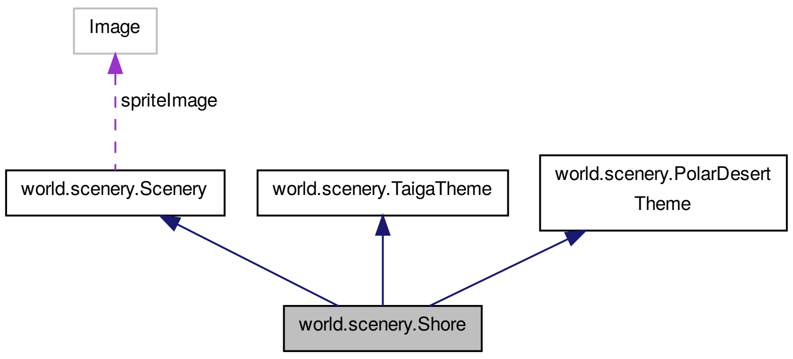
7.25 world.scenery.Shore Class Reference

Class used to handle a shore object.

Inheritance diagram for world.scenery.Shore:



Collaboration diagram for world.scenery.Shore:



Public Member Functions

- Shore (final String type, final String orientation)

    Shore class constructor.

Additional Inherited Members

7.25.1 Detailed Description

Class used to handle a shore object.

Definition at line 12 of file Shore.java.

7.25.2 Constructor & Destructor Documentation

7.25.2.1 world.scenery.Shore.Shore (final String *type,* final String *orientation*)

Shore class constructor.

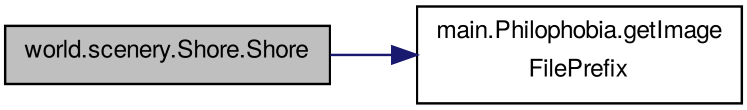
Parameters

type	Style of the shore
orientation	Orientation of the sprite (possible : "n", "s", "e", "w", "nee", "nwe", "nei", "nwi", "see", "swe", "sei", "swi")

Definition at line 19 of file Shore.java.

```
                                                           {

        super(Philophobia.getImageFilePrefix() + type + "shore" + orientation + ".png");

    }
```

Here is the call graph for this function:



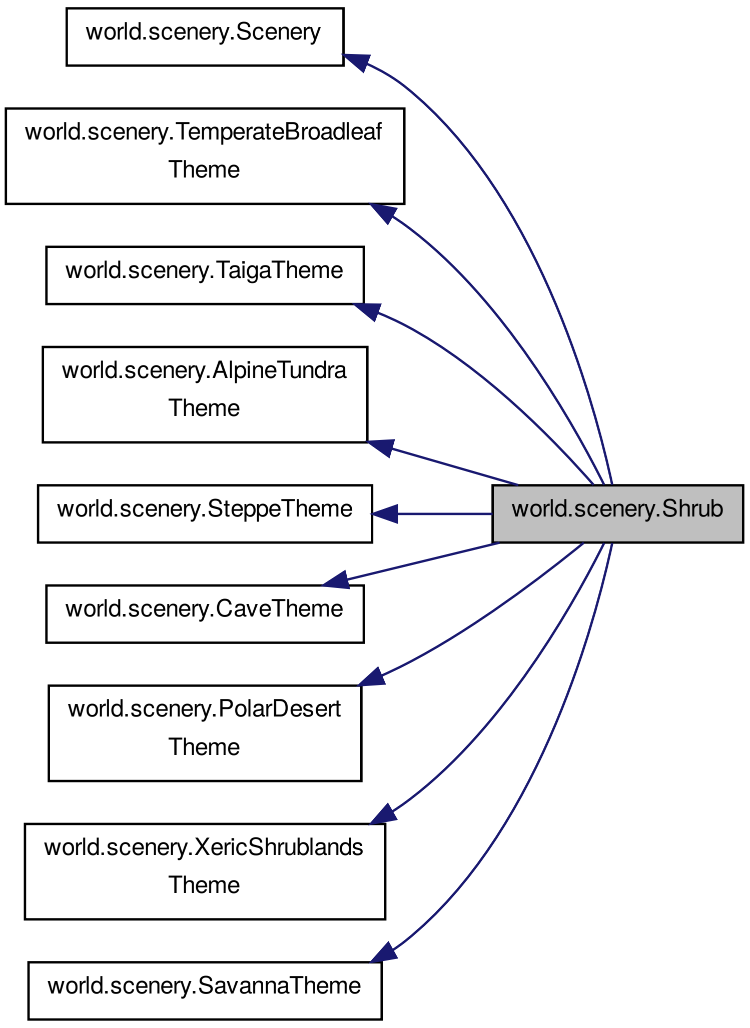
The documentation for this class was generated from the following file:

- world/scenery/Shore.java

7.26 world.scenery.Shrub Class Reference

Class used to handle a shrub object.

Inheritance diagram for world.scenery.Shrub:


world.scenery.Scenery
world.scenery.TemperateBroadleaf Theme
world.scenery.TaigaTheme
world.scenery.AlpineTundra Theme
world.scenery.SteppeTheme
world.scenery.Shrub
world.scenery.CaveTheme
world.scenery.PolarDesert Theme
world.scenery.XericShrublands Theme
world.scenery.SavannaTheme

Collaboration diagram for world.scenery.Shrub:

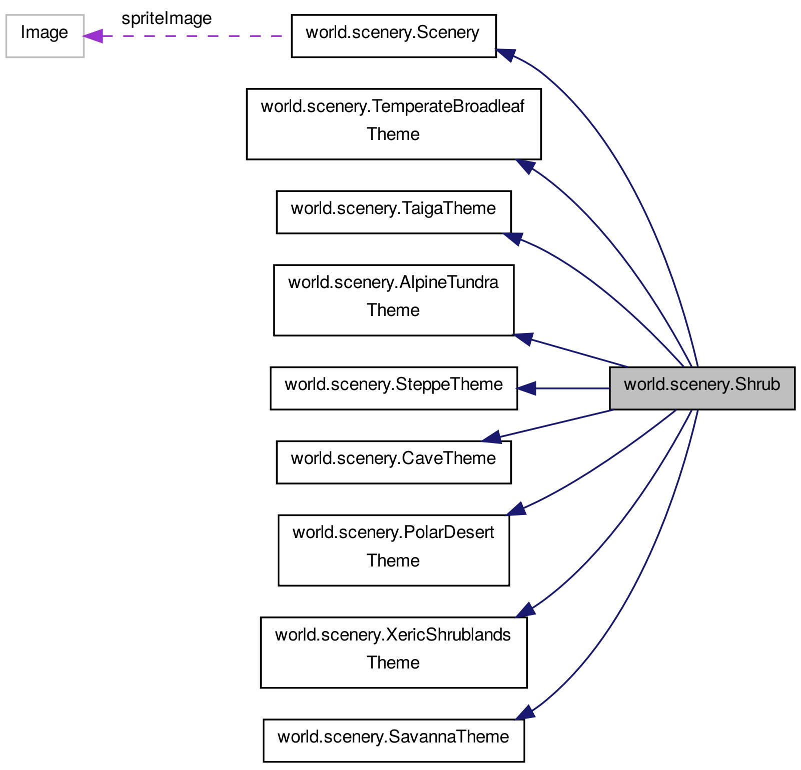


Image
spriteImage
world.scenery.Scenery
world.scenery.TemperateBroadleaf Theme
world.scenery.TaigaTheme
world.scenery.AlpineTundra Theme
world.scenery.SteppeTheme
world.scenery.CaveTheme
world.scenery.PolarDesert Theme
world.scenery.XericShrublands Theme
world.scenery.SavannaTheme
world.scenery.Shrub

Public Member Functions

- Shrub (final String type)

    Shrub class constructor.

Additional Inherited Members

7.26.1 Detailed Description

Class used to handle a shrub object.

Definition at line 18 of file Shrub.java.

7.26.2 Constructor & Destructor Documentation

7.26.2.1 world.scenery.Shrub.Shrub (final String *type*)

Shrub class constructor.

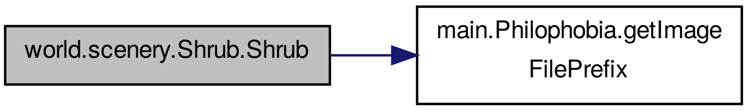
Parameters

type	Style of the shrub

Definition at line 24 of file Shrub.java.

```
                                   {

        super(Philophobia.getImageFilePrefix() + type + "shrub.png");

    }
```

Here is the call graph for this function:


world.scenery.Shrub.Shrub
main.Philophobia.getImage FilePrefix

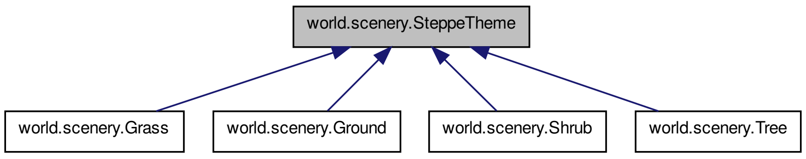
The documentation for this class was generated from the following file:

- world/scenery/Shrub.java

7.27 world.scenery.SteppeTheme Interface Reference

Interface used to define that a Scenery object have a Steppe style.

Inheritance diagram for world.scenery.SteppeTheme:



Static Public Attributes

- static String STEPPE = "steppe"

  String used to tell in which file is the sprite matching the Steppe style.

7.27.1 Detailed Description

Interface used to define that a Scenery object have a Steppe style.

Definition at line 7 of file SteppeTheme.java.

7.27.2 Member Data Documentation

7.27.2.1 String world.scenery.SteppeTheme.STEPPE = "steppe" [static]

String used to tell in which file is the sprite matching the Steppe style.

Definition at line 13 of file SteppeTheme.java.

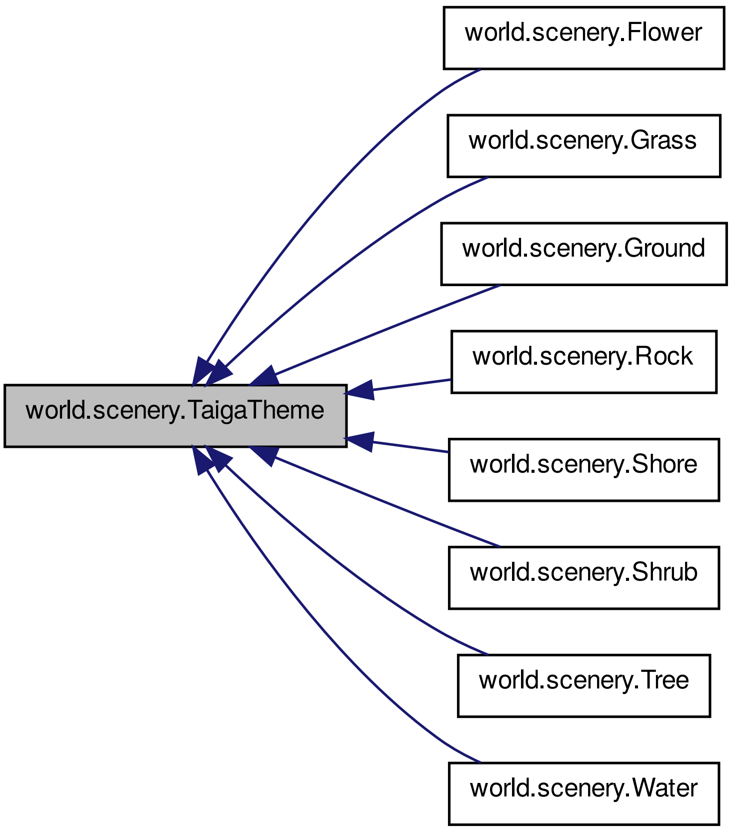
The documentation for this interface was generated from the following file:

- world/scenery/SteppeTheme.java

7.28 world.scenery.TaigaTheme Interface Reference

Interface used to define that a Scenery object have a Taiga style.

Inheritance diagram for world.scenery.TaigaTheme:



Static Public Attributes

- static String TAIGA = "taiga"

  String used to tell in which file is the sprite matching the Taiga style.

7.28.1 Detailed Description

Interface used to define that a Scenery object have a Taiga style.

Definition at line 7 of file TaigaTheme.java.

7.28.2 Member Data Documentation

7.28.2.1 String world.scenery.TaigaTheme.TAIGA = "taiga" [static]

String used to tell in which file is the sprite matching the Taiga style.

Definition at line 13 of file TaigaTheme.java.

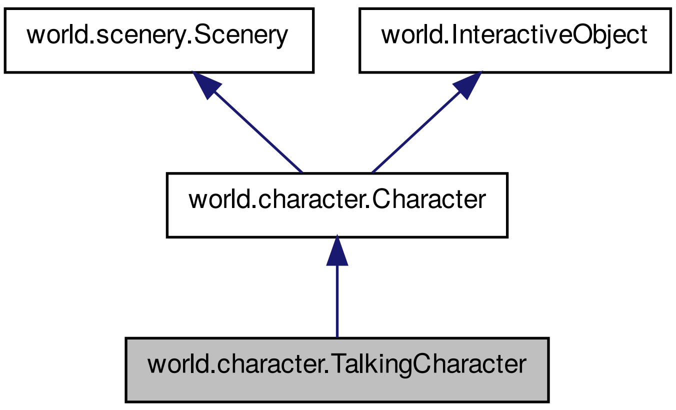
The documentation for this interface was generated from the following file:

- world/scenery/TaigaTheme.java

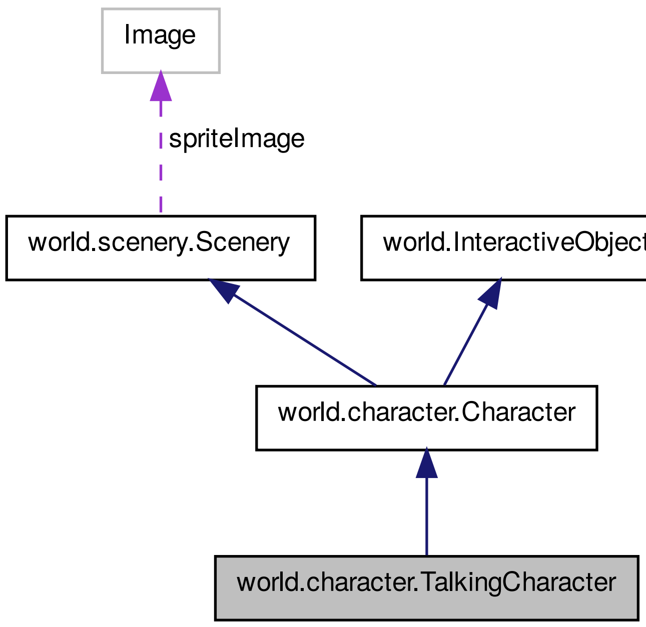
7.29 world.character.TalkingCharacter Class Reference

Class handling a talking character.

Inheritance diagram for world.character.TalkingCharacter:



Collaboration diagram for world.character.TalkingCharacter:



Public Member Functions

- TalkingCharacter (final String imagePath)

  TalkingCharacter constructor.
- void talk ()

  Method making the character talk.
- void triggerAction ()

  Method to be called when the player is near the object and presses the action key.

Additional Inherited Members

7.29.1 Detailed Description

Class handling a talking character.

Definition at line 9 of file TalkingCharacter.java.

7.29.2 Constructor & Destructor Documentation

7.29.2.1 world.character.TalkingCharacter.TalkingCharacter (final String *imagePath*)

TalkingCharacter constructor.

Parameters

imagePath	Path of the image representing the character

Definition at line 15 of file TalkingCharacter.java.

```
                                            {
        super(imagePath);
    }
```

7.29.3 Member Function Documentation

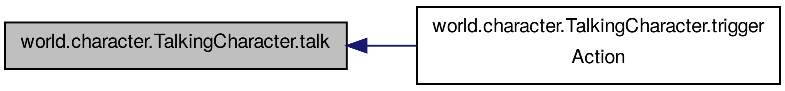
7.29.3.1 void world.character.TalkingCharacter.talk ( )

Method making the character talk.

Definition at line 22 of file TalkingCharacter.java.

```
                           {

    }
```

Here is the caller graph for this function:


world.character.TalkingCharacter.talk
world.character.TalkingCharacter.trigger Action

7.29.3.2 void world.character.TalkingCharacter.triggerAction ( )

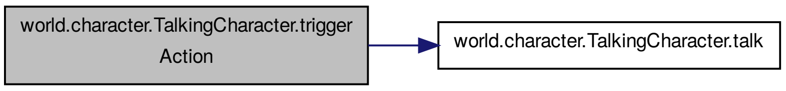
Method to be called when the player is near the object and presses the action key.

Implements world.InteractiveObject.

Definition at line 30 of file TalkingCharacter.java.

```
                               {
        talk();
    }
```

Here is the call graph for this function:


world.character.TalkingCharacter.trigger Action
world.character.TalkingCharacter.talk

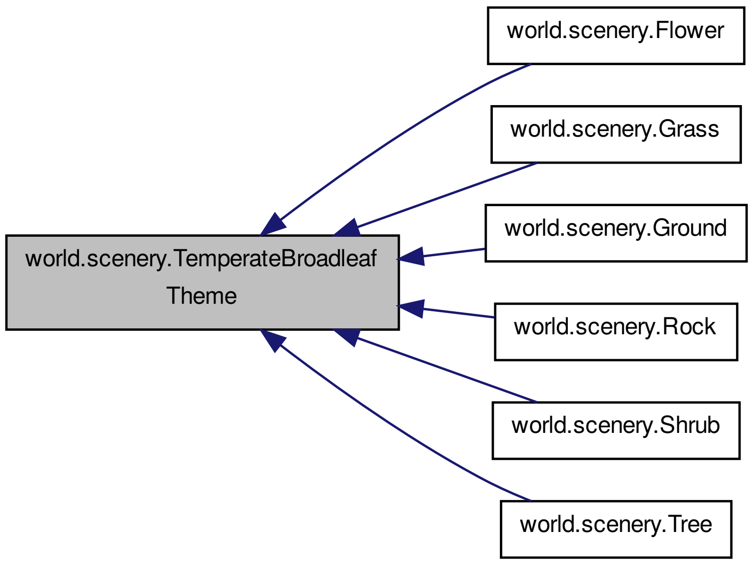
The documentation for this class was generated from the following file:

- world/character/TalkingCharacter.java

7.30 world.scenery.TemperateBroadleafTheme Interface Reference

Interface used to define that a Scenery object have a Temperate Broadleaf style.

Inheritance diagram for world.scenery.TemperateBroadleafTheme:



Static Public Attributes

- static String TEMPERATE_BROADLEAF = "temperatebroadleaf"

  String used to tell in which file is the sprite matching the Temperate Broadleaf style.

7.30.1 Detailed Description

Interface used to define that a Scenery object have a Temperate Broadleaf style.

Definition at line 7 of file TemperateBroadleafTheme.java.

7.30.2 Member Data Documentation

7.30.2.1 String world.scenery.TemperateBroadleafTheme.TEMPERATE_BROADLEAF = "temperatebroadleaf" [static]

String used to tell in which file is the sprite matching the Temperate Broadleaf style.

Definition at line 13 of file TemperateBroadleafTheme.java.

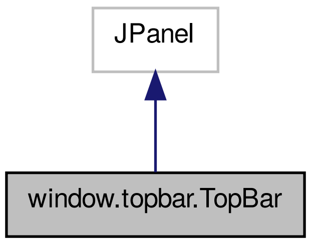
The documentation for this interface was generated from the following file:

- world/scenery/TemperateBroadleafTheme.java

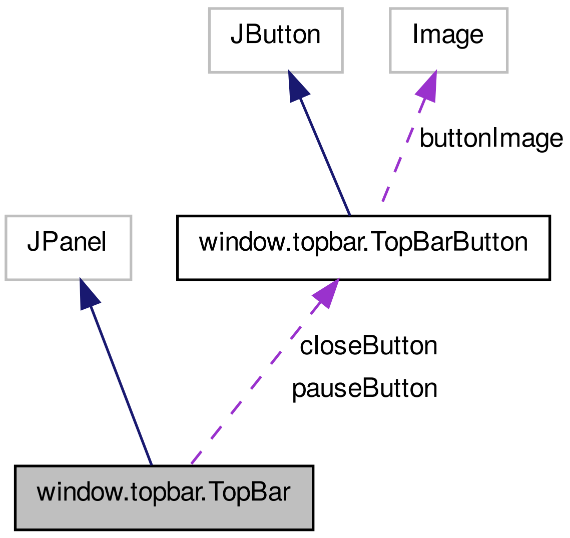
7.31 window.topbar.TopBar Class Reference

Class handling the top bar of the program's main window.

Inheritance diagram for window.topbar.TopBar:



Collaboration diagram for window.topbar.TopBar:



Public Member Functions

- TopBar ()

  Constructor of the TopBar class.
- void hidePauseButton ()
- void showPauseButton ()
- void paintComponent (Graphics g)

Static Public Member Functions

- static int getTopBarHeight ()
- static int getTopBarMargin ()

Protected Attributes

- TopBarButton closeButton

  Button closing the program when activated.
- TopBarButton pauseButton

  Button pausing the program when activated.

Static Protected Attributes

- static int TOPBAR_HEIGHT = 42

  Height of the UI's top bar (default: 42px)
- static int TOPBAR_MARGIN = 10

  Margin of the top bar (default: 10px)

7.31.1 Detailed Description

Class handling the top bar of the program's main window.

Definition at line 16 of file TopBar.java.

7.31.2 Constructor & Destructor Documentation

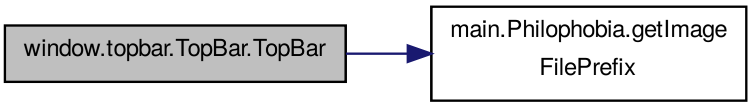
7.31.2.1 window.topbar.TopBar.TopBar ( )

Constructor of the TopBar class.

Definition at line 41 of file TopBar.java.

```
41                    {
42
43         super();
44
45         Philophobia.getVerbose().calls("Creating TopBar class", "window/topbar/TopBar.java", "
       TopBar.TopBar()");
46
47         this.setLayout(new BoxLayout(this, BoxLayout.LINE_AXIS));
48         this.add(Box.createHorizontalGlue());
49
50         closeButton = new TopBarButton(Philophobia.getImageFilePrefix() + "closebutton.png");
51         closeButton.setPreferredSize(new Dimension(TOPBAR_HEIGHT -
       TOPBAR_MARGIN, TOPBAR_HEIGHT - TOPBAR_MARGIN));
52         closeButton.setMinimumSize(new Dimension(TOPBAR_HEIGHT -
       TOPBAR_MARGIN, TOPBAR_HEIGHT - TOPBAR_MARGIN));
53         closeButton.setMaximumSize(new Dimension(TOPBAR_HEIGHT -
       TOPBAR_MARGIN, TOPBAR_HEIGHT - TOPBAR_MARGIN));
54
55         pauseButton = new TopBarButton(Philophobia.getImageFilePrefix() + "pausebutton.png");
56         pauseButton.setPreferredSize(new Dimension(TOPBAR_HEIGHT -
       TOPBAR_MARGIN, TOPBAR_HEIGHT - TOPBAR_MARGIN));
57         pauseButton.setMinimumSize(new Dimension(TOPBAR_HEIGHT -
       TOPBAR_MARGIN, TOPBAR_HEIGHT - TOPBAR_MARGIN));
58         pauseButton.setMaximumSize(new Dimension(TOPBAR_HEIGHT -
       TOPBAR_MARGIN, TOPBAR_HEIGHT - TOPBAR_MARGIN));
59
60         this.add(closeButton);
61     }
```

Here is the call graph for this function:


window.topbar.TopBar.TopBar
main.Philophobia.getImage FilePrefix

7.31.3 Member Function Documentation

7.31.3.1 static int window.topbar.TopBar.getTopBarHeight ( ) [static]

Definition at line 63 of file TopBar.java.

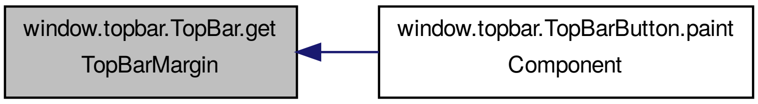
```
                                      {
        return TOPBAR_HEIGHT;
    }
```

7.31.3.2 static int window.topbar.TopBar.getTopBarMargin ( ) [static]

Definition at line 67 of file TopBar.java.

```
                                      {
        return TOPBAR_MARGIN;
    }
```

Here is the caller graph for this function:


window.topbar.TopBar.get TopBarMargin
window.topbar.TopBarButton.paint Component

7.31.3.3 void window.topbar.TopBar.hidePauseButton ( )

Definition at line 71 of file TopBar.java.

```
                                 {
        Philophobia.getVerbose().calls("Hidding pause button", "window/topbar/TopBar.java", "
    TopBar.hidePauseButton()");
        this.removeAll();
        this.add(Box.createHorizontalGlue());
        this.add(closeButton);
        this.repaint();
    }
```

7.31.3.4 void window.topbar.TopBar.paintComponent (Graphics *g*)

Definition at line 89 of file TopBar.java.

```
                                    {
        Philophobia.getVerbose().calls("Paintint TopBar component", "window/topbar/TopBar.java", "
    TopBar.painComponent(Graphics)");
        // Top bar background
        g.fillRect(0, 0, this.getWidth(), this.getHeight());

        super.paintComponents(g);
    }
```

7.31.3.5 void window.topbar.TopBar.showPauseButton ( )

Definition at line 79 of file TopBar.java.

```
                             {
        Philophobia.getVerbose().calls("Showing pause button", "window/topbar/TopBar.java", "
    TopBar.showPauseButton()");
        this.removeAll();
        this.add(Box.createHorizontalGlue());
        this.add(pauseButton);
        this.add(Box.createRigidArea(new Dimension(TOPBAR_MARGIN, 0)));
        this.add(closeButton);
        this.repaint();
    }
```

7.31.4 Member Data Documentation

7.31.4.1 TopBarButton window.topbar.TopBar.closeButton [protected]

Button closing the program when activated.

Definition at line 31 of file TopBar.java.

7.31.4.2 TopBarButton window.topbar.TopBar.pauseButton [protected]

Button pausing the program when activated.

Definition at line 36 of file TopBar.java.

7.31.4.3 int window.topbar.TopBar.TOPBAR_HEIGHT = 42 [static], [protected]

Height of the UI's top bar (default: 42px)

Definition at line 21 of file TopBar.java.

7.31.4.4 int window.topbar.TopBar.TOPBAR_MARGIN = 10 [static], [protected]

Margin of the top bar (default: 10px)

Definition at line 26 of file TopBar.java.

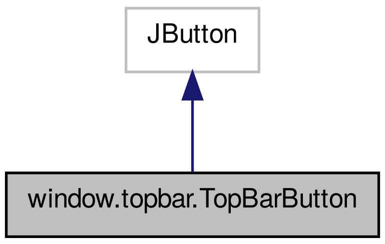
The documentation for this class was generated from the following file:

- window/topbar/TopBar.java

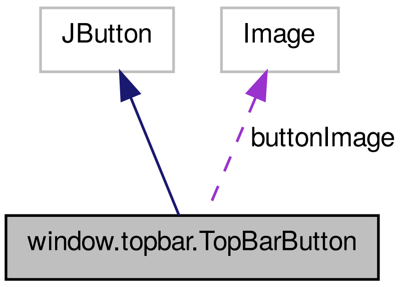
7.32 window.topbar.TopBarButton Class Reference

Class handling the UI buttons.

Inheritance diagram for window.topbar.TopBarButton:



Collaboration diagram for window.topbar.TopBarButton:



Public Member Functions

- TopBarButton (String imagePath)

    UIButton constructor.
- void paintComponent (Graphics g)

Protected Attributes

- Image buttonImage

    Image of the graphical representation of the button.

7.32.1 Detailed Description

Class handling the UI buttons.

Definition at line 19 of file TopBarButton.java.

7.32.2 Constructor & Destructor Documentation

7.32.2.1 window.topbar.TopBarButton.TopBarButton (String *imagePath*)

UIButton constructor.

Parameters

imagePath	path of the button's image

See Also

buttonImage

Definition at line 31 of file TopBarButton.java.

```
                                        {

        super();

        Philophobia.getVerbose().calls("Creating TopBarButton class", "window/ui/TopBarButton.java", "
    TopBarButton.TopBarButton(String)");

        try {
            buttonImage = ImageIO.read(new File(imagePath));
            this.setIcon(new ImageIcon(buttonImage));
            this.setBorder(null);
        } catch(IOException e) {
            Philophobia.getVerbose().warning("Button image load failed: " + e.getMessage(), "
    window/ui/TopBarButton.java", "TopBarButton.TopBarButton(String)");
            buttonImage = null;
        }

    }
```

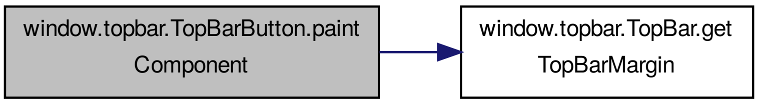
7.32.3 Member Function Documentation

7.32.3.1 void window.topbar.TopBarButton.paintComponent (Graphics *g*)

Definition at line 48 of file TopBarButton.java.

```
                                         {
        Philophobia.getVerbose().calls("painting TopBarButton component", "window/ui/TopBarButton.java", "
    TopBarButton.paintComponent(Graphics)");

        int buttonSize = TopBar.getTopBarHeight() - TopBar.getTopBarMargin();

        g.drawImage(buttonImage, 0, 0, this.getWidth(), this.getHeight(), this);
    }
```

Here is the call graph for this function:


window.topbar.TopBarButton.paint Component
window.topbar.TopBar.get TopBarMargin

7.32.4 Member Data Documentation

7.32.4.1 Image window.topbar.TopBarButton.buttonImage [protected]

Image of the graphical representation of the button.

Definition at line 24 of file TopBarButton.java.

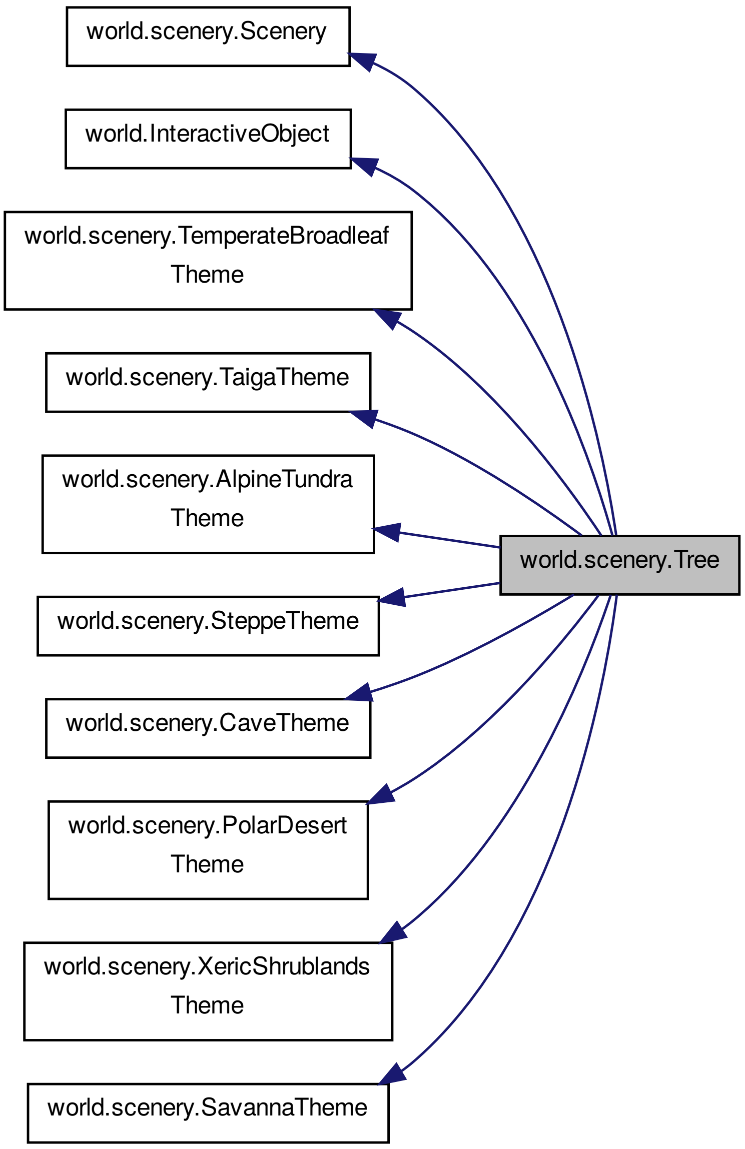
The documentation for this class was generated from the following file:

- window/topbar/TopBarButton.java

7.33 world.scenery.Tree Class Reference

Class used to handle a tree object.

Inheritance diagram for world.scenery.Tree:


world.scenery.Scenery
world.InteractiveObject
world.scenery.TemperateBroadleaf Theme
world.scenery.TaigaTheme
world.scenery.AlpineTundra Theme
world.scenery.SteppeTheme
world.scenery.CaveTheme
world.scenery.PolarDesert Theme
world.scenery.XericShrublands Theme
world.scenery.SavannaTheme
world.scenery.Tree

Collaboration diagram for world.scenery.Tree:

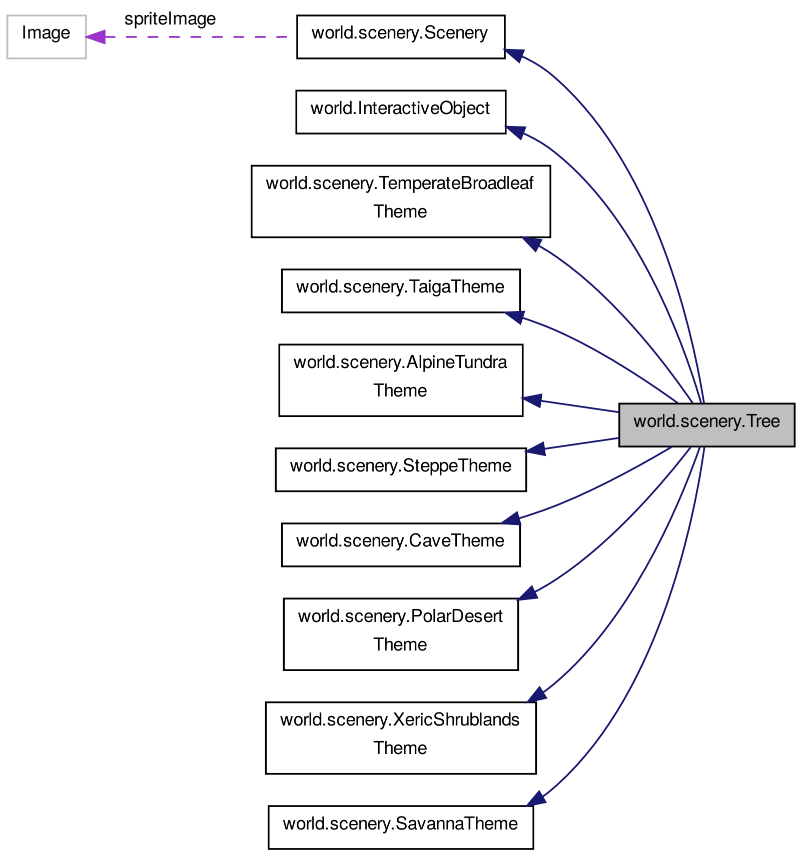


Image
spriteImage
world.scenery.Scenery
world.InteractiveObject
world.scenery.TemperateBroadleaf Theme
world.scenery.TaigaTheme
world.scenery.AlpineTundra Theme
world.scenery.Tree
world.scenery.SteppeTheme
world.scenery.CaveTheme
world.scenery.PolarDesert Theme
world.scenery.XericShrublands Theme
world.scenery.SavannaTheme

Public Member Functions

- Tree (final String type)

    Tree class constructor.
- void triggerAction ()

    Method to be called when the player is near the object and presses the action key.
- void playerNear ()

    Method to be called when the player is near the object.
- void playerEnter ()

    Method to be called when the player was near and is now over the object.
- void playerLeave ()

    Method to be called when the player was over and is now near the object.
- boolean isTraversable ()

    Returns true if the player is able to pass through the object and false if the player is not able to cross the object.

Additional Inherited Members

7.33.1 Detailed Description

Class used to handle a tree object.

Definition at line 19 of file Tree.java.

7.33.2 Constructor & Destructor Documentation

7.33.2.1 world.scenery.Tree.Tree (final String *type*)

Tree class constructor.

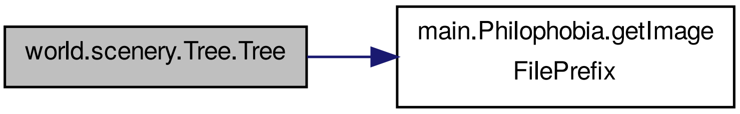
Parameters

type	Style of the tree

Definition at line 25 of file Tree.java.

```
                                    {

        super(Philophobia.getImageFilePrefix() + type + "tree.png", 96, Scenery.getSceneryWidth(), 0, -48);

    }
```

Here is the call graph for this function:



7.33.3 Member Function Documentation

7.33.3.1 boolean world.scenery.Tree.isTraversable ( )

Returns true if the player is able to pass through the object and false if the player is not able to cross the object.

Implements world.InteractiveObject.

Definition at line 51 of file Tree.java.

```
                                    {
        return false;
    }
```

7.33.3.2 void world.scenery.Tree.playerEnter ( )

Method to be called when the player was near and is now over the object.

Implements world.InteractiveObject.

Definition at line 43 of file Tree.java.

```
                               {

    }
```

7.33.3.3 void world.scenery.Tree.playerLeave ( )

Method to be called when the player was over and is now near the object.

Implements world.InteractiveObject.

Definition at line 47 of file Tree.java.

```
                                  {
    }
```

7.33.3.4 void world.scenery.Tree.playerNear ( )

Method to be called when the player is near the object.

Implements world.InteractiveObject.

Definition at line 39 of file Tree.java.

```
                                 {
    }
```

7.33.3.5 void world.scenery.Tree.triggerAction ( )

Method to be called when the player is near the object and presses the action key.

Implements world.InteractiveObject.

Definition at line 35 of file Tree.java.

```
                                    {
    }
```

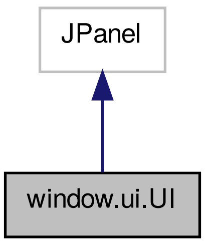
The documentation for this class was generated from the following file:

- world/scenery/Tree.java

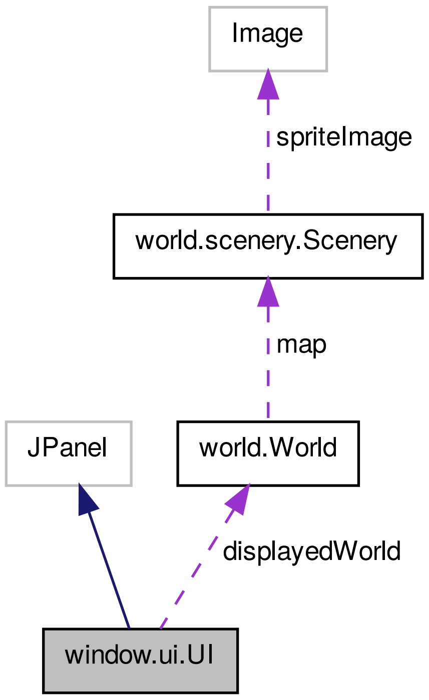
7.34 window.ui.UI Class Reference

Class used to handle the window's user interface.

Inheritance diagram for window.ui.UI:



Collaboration diagram for window.ui.UI:



Public Member Functions

- UI (int windowHeight, int windowWidth)

  UI class constructor.
- UI (int windowHeight, int windowWidth, World world)

  UI class constructor.
- void setDisplayedWorld (World world)

  Setter for the displayedWorld field.
- void paintComponent (Graphics g)

  Function called when the program ask to paint the graphics.

Protected Attributes

- World displayedWorld

  World displayed inside the window.
- int windowHeight

  Window height.
- int windowWidth

  Window width.

Private Member Functions

- void drawWorld (Graphics g)

  Function called to draw the world.

7.34.1 Detailed Description

Class used to handle the window's user interface.

This class can be in different states

Definition at line 16 of file UI.java.

7.34.2 Constructor & Destructor Documentation

7.34.2.1 window.ui.UI.UI (int *windowHeight*, int *windowWidth*)

UI class constructor.

Parameters

windowHeight	Height of the window
windowWidth	Width of the window

Definition at line 38 of file UI.java.

```
                                                    {
        // We call the parent's constructor
        super();

        Philophobia.getVerbose().calls("Creating UI class", "window/ui/UI.java", "UI.UI(int, int)");

        this.windowHeight = windowHeight;
        this.windowWidth = windowWidth;

    }
```

7.34.2.2 window.ui.UI.UI (int *windowHeight*, int *windowWidth*, World *world*)

UI class constructor.

Parameters

windowHeight	Height of the window
windowWidth	Width of the window
world	World to display

Definition at line 55 of file UI.java.

```
                                                                 {
        super();

        Philophobia.getVerbose().calls("Creating UI class", "window/ui/UI.java", "UI.UI(int, int,World)");

        this.windowHeight = windowHeight;
        this.windowWidth = windowWidth;
        this.displayedWorld = world;
    }
```

7.34.3 Member Function Documentation

7.34.3.1 void window.ui.UI.drawWorld (Graphics *g*) [private]

Function called to draw the world.

This function just display each Scenery one by one but trees are displayed after everything else as they must be on top of everything because of their size

See Also

displayedWorld

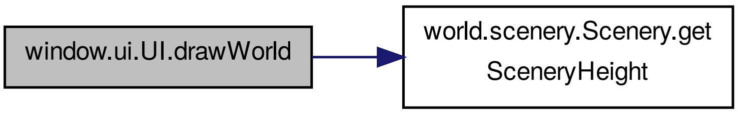
Parameters

g	Graphics to use

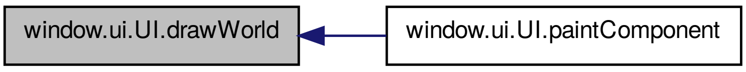
Definition at line 94 of file UI.java.

```
                                      {

       Philophobia.getVerbose().calls("Painting the world", "window/ui/UI.java", "UI.drawWorld(Graphics)")
;

       Scenery[][] worldMap = displayedWorld.getMap();
       int worldMapXSize = displayedWorld.getSizeX();
        int worldMapYSize = displayedWorld.getSizeY();

        for(int i = 0 ; i < worldMapXSize ; ++i) {
            for(int j = 0 ; j < worldMapYSize ; ++j) {
                if(!worldMap[i][j].getClass().getName().equals("Tree"))
                    worldMap[i][j].drawScenery(g, i*Scenery.getSceneryHeight(), j*Scenery.getSceneryWidth()
, this);
            }
        }

        for(int i = 0 ; i < worldMapXSize ; ++i) {
            for(int j = 0 ; j < worldMapYSize ; ++j) {
                if(worldMap[i][j].getClass().getName().equals("Tree"))
                    worldMap[i][j].drawScenery(g, i*Scenery.getSceneryHeight(), j*Scenery.getSceneryWidth()
, this);
            }
        }

    }
```

Here is the call graph for this function:



Here is the caller graph for this function:



7.34.3.2 void window.ui.UI.paintComponent (Graphics *g*)

Function called when the program ask to paint the graphics.

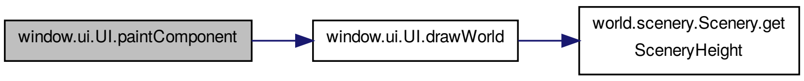
Parameters

g	Graphics to use

Definition at line 78 of file UI.java.

```
                                        {
        Philophobia.getVerbose().calls("Painting UI components", "window/ui/UI.java", "
    UI.paintComponent(Graphics)");
        super.paintComponent(g);
        drawWorld(g);
    }
```

Here is the call graph for this function:


window.ui.UI.paintComponent
window.ui.UI.drawWorld
world.scenery.Scenery.get SceneryHeight

7.34.3.3 void window.ui.UI.setDisplayedWorld (World *world*)

Setter for the displayedWorld field.

See Also

displayedWorld

Parameters

world	World to display

Definition at line 70 of file UI.java.

```
                                           {
        this.displayedWorld = world;
    }
```

7.34.4 Member Data Documentation

7.34.4.1 World window.ui.UI.displayedWorld [protected]

World displayed inside the window.

Definition at line 21 of file UI.java.

7.34.4.2 int window.ui.UI.windowHeight [protected]

Window height.

Definition at line 26 of file UI.java.

7.34.4.3 int window.ui.UI.windowWidth [protected]

Window width.

Definition at line 31 of file UI.java.

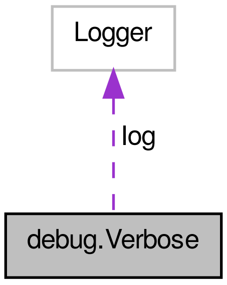
The documentation for this class was generated from the following file:

- window/ui/UI.java

7.35 debug.Verbose Class Reference

Class used to display and log messages all over this program.

Collaboration diagram for debug.Verbose:



Public Member Functions

- Verbose (int level)

  Class constructor.
- void critical (String message, String file, String location)

  Display and log a message with additionnal informations in the log file The message is displayed and logged if and only if the verbose level is greater or equal to 1.
- void serious (String message, String file, String location)

  Display and log a message with additionnal informations in the log file The message is displayed and logged if and only if the verbose level is greater or equal to 2.
- void warning (String message, String file, String location)

  Display and log a message with additionnal informations in the log file The message is displayed and logged if and only if the verbose level is greater or equal to 3.
- void information (String message, String file, String location)

  Display and log a message with additionnal informations in the log file The message is displayed and logged if and only if the verbose level is greater or equal to 4.
- void calls (String message, String file, String location)

  Display and log a message with additionnal informations in the log file The message is displayed and logged if and only if the verbose level is equal to 5.

Protected Attributes

- int verboseMode = 0

  Level of verbosing, from 0 to 5.

Package Attributes

- Logger log

    Logger used to log messages into a file.

7.35.1 Detailed Description

Class used to display and log messages all over this program.

Definition at line 11 of file Verbose.java.

7.35.2 Constructor & Destructor Documentation

7.35.2.1 debug.Verbose.Verbose (int *level*)

Class constructor.

Parameters

level	Level of "verbosing"

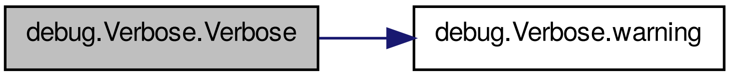
See Also

verboseMode

Definition at line 36 of file Verbose.java.

```
                            {
        this.verboseMode = level;

        if(level > 0) {
            System.out.println("Verbose mode activated at level " + level);

                log = Logger.getLogger("Philophobia.log");

                try {
                    log.addHandler(new FileHandler("Philophobia.log"));
                } catch (IOException e) {
                    warning("Error initializing the log file", "org/debug/Verbose.java", "
Verbose.Verbose(int)");
                }

                log.setLevel(Level.parse("ALL"));

                log.info("Verbose at level " + level + ".");
                log.info("Displayed messages are :");
                log.info("- Criticals");

                if(level >= 2) {
                    log.info("- Serious");
                }
                if(level >= 3) {
                    log.info("- Warnings");
                }
                if(level >= 4) {
                    log.info("- Informations");
                }
                if(level >= 5) {
                    log.info("- Class instanciations and function calls");
                }
        }

        this.calls("Verbose class created", "org/debug/Verbose.java", "Verbose.Verbose(int)");
    }
```

Here is the call graph for this function:



7.35.3 Member Function Documentation

7.35.3.1 void debug.Verbose.calls (String *message*, String *file*, String *location*)

Display and log a message with additionnal informations in the log file The message is displayed and logged if and only if the verbose level is equal to 5.

Parameters

message	Message content
file	File in which this function is called
location	Class and function in which this function is called

Definition at line 145 of file Verbose.java.

```
                                                                          {
        if(verboseMode >= 5) {
            System.out.println("Calls : " + message);

            log.finer("Calls : " + message + " in file " + file + " in " + location);
        }
    }
```

7.35.3.2 void debug.Verbose.critical (String *message*, String *file*, String *location*)

Display and log a message with additionnal informations in the log file The message is displayed and logged if and only if the verbose level is greater or equal to 1.

Parameters

message	Message content
file	File in which this function is called
location	Class and function in which this function is called

Definition at line 81 of file Verbose.java.

```
                                                                             {
        if(verboseMode >= 1) {
            System.out.println("Critical : " + message);

            log.severe("Critical : " + message + " in file " + file + " in " + location);
        }
    }
```

7.35.3.3 void debug.Verbose.information (String *message*, String *file*, String *location*)

Display and log a message with additionnal informations in the log file The message is displayed and logged if and only if the verbose level is greater or equal to 4.

Parameters

message	Message content
file	File in which this function is called
location	Class and function in which this function is called

Definition at line 129 of file Verbose.java.

```
                                                                            {
        if(verboseMode >= 4) {
            System.out.println("Information : " + message);

            log.info("Information : " + message + " in file " + file + " in " + location);
        }
    }
```

7.35.3.4 void debug.Verbose.serious (String *message*, String *file*, String *location*)

Display and log a message with additionnal informations in the log file The message is displayed and logged if and only if the verbose level is greater or equal to 2.

Parameters

message	Message content
file	File in which this function is called
location	Class and function in which this function is called

Definition at line 97 of file Verbose.java.

```
                                                                            {
        if(verboseMode >= 2) {
            System.out.println("Serious : " + message);

            log.severe("Serious : " + message + " in file " + file + " in " + location);
        }
    }
```

7.35.3.5 void debug.Verbose.warning (String *message*, String *file*, String *location*)

Display and log a message with additionnal informations in the log file The message is displayed and logged if and only if the verbose level is greater or equal to 3.

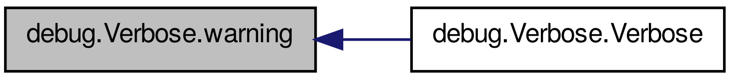
Parameters

message	Message content
file	File in which this function is called
location	Class and function in which this function is called

Definition at line 113 of file Verbose.java.

```
                                                                            {
        if(verboseMode >= 3) {
            System.out.println("Warning : " + message);

            log.warning("Warning : " + message + " in file " + file + " in " + location);
        }
    }
```

Here is the caller graph for this function:



7.35.4 Member Data Documentation

7.35.4.1 Logger debug.Verbose.log [package]

Logger used to log messages into a file.

Definition at line 28 of file Verbose.java.

7.35.4.2 int debug.Verbose.verboseMode = 0 [protected]

Level of verbosing, from 0 to 5.

The number of information displayed and logged goes increasingly as the number increase

- Level 1 : Only critical errors and exeptions are displayed on the console
- Level 2 : Serious but not critical errors and level 1 displayed
- Level 3 : Warning and level 2 displayed
- Level 4 : Informations and level 3 displayed
- Level 5 : Everything, included Class instanciations and function calls and level 4 displayed

Definition at line 23 of file Verbose.java.

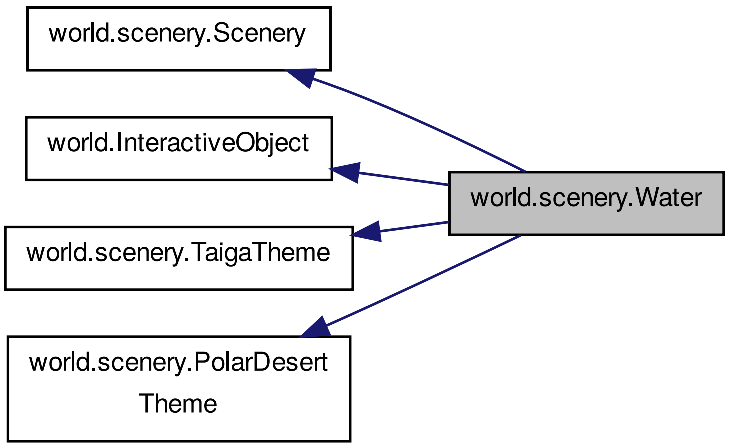
The documentation for this class was generated from the following file:

- debug/Verbose.java

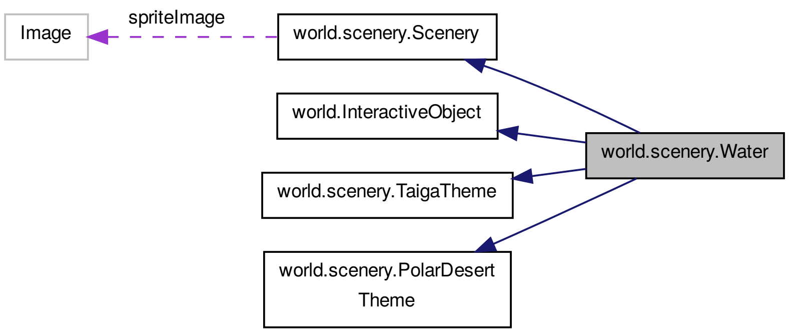
7.36 world.scenery.Water Class Reference

Class used to handle a water object.

Inheritance diagram for world.scenery.Water:



Collaboration diagram for world.scenery.Water:



Public Member Functions

- Water (final String type)

  Water class constructor.
- void triggerAction ()

  Method to be called when the player is near the object and presses the action key.
- void playerNear ()

  Method to be called when the player is near the object.
- void playerEnter ()

  Method to be called when the player was near and is now over the object.
- void playerLeave ()

  Method to be called when the player was over and is now near the object.
- boolean isTraversable ()

Returns true if the player is able to pass through the object and false if the player is not able to cross the object.

Additional Inherited Members

7.36.1 Detailed Description

Class used to handle a water object.

Definition at line 13 of file Water.java.

7.36.2 Constructor & Destructor Documentation

7.36.2.1 world.scenery.Water.Water (final String *type*)

Water class constructor.

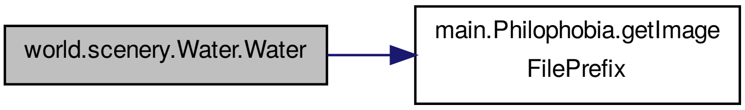
Parameters

type	Style of the water

Definition at line 19 of file Water.java.

```
                                    {

        super(Philophobia.getImageFilePrefix() + type + "water.png");

    }
```

Here is the call graph for this function:


world.scenery.Water.Water
main.Philophobia.getImage FilePrefix

7.36.3 Member Function Documentation

7.36.3.1 boolean world.scenery.Water.isTraversable ( )

Returns true if the player is able to pass through the object and false if the player is not able to cross the object.

Implements world.InteractiveObject.

Definition at line 45 of file Water.java.

```
                                   {
        return false;
    }
```

7.36.3.2 void world.scenery.Water.playerEnter ( )

Method to be called when the player was near and is now over the object.

Implements world.InteractiveObject.

Definition at line 37 of file Water.java.

```
                                   {
    }
```

7.36.3.3 void world.scenery.Water.playerLeave ( )

Method to be called when the player was over and is now near the object.

Implements world.InteractiveObject.

Definition at line 41 of file Water.java.

```
                                   {
    }
```

7.36.3.4 void world.scenery.Water.playerNear ( )

Method to be called when the player is near the object.

Implements world.InteractiveObject.

Definition at line 33 of file Water.java.

```
                                  {
        // AI talking
    }
```

7.36.3.5 void world.scenery.Water.triggerAction ( )

Method to be called when the player is near the object and presses the action key.

Implements world.InteractiveObject.

Definition at line 29 of file Water.java.

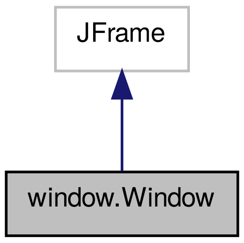
```
                                     {
    }
```

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

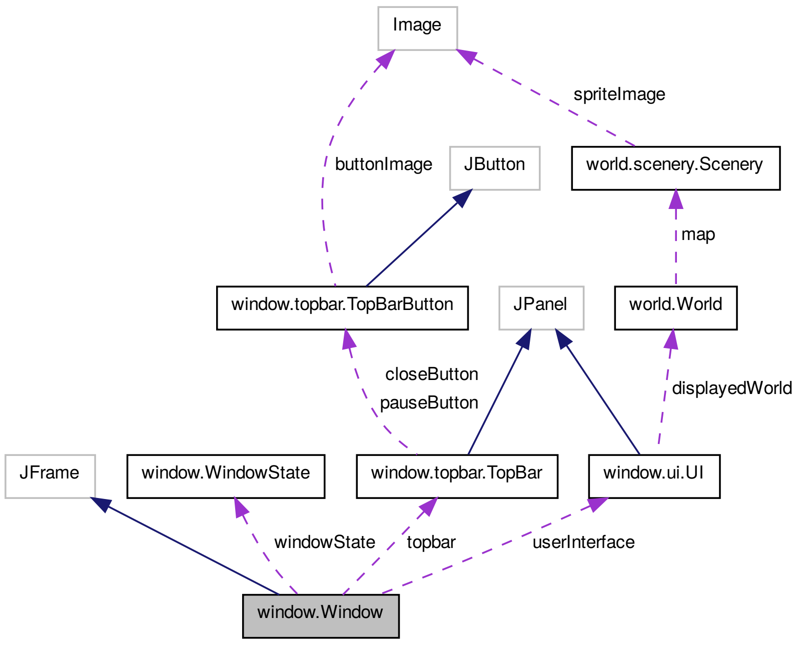
- world/scenery/Water.java

7.37 window.Window Class Reference

Inheritance diagram for window.Window:



Collaboration diagram for window.Window:



Public Member Functions

- Window ()

  Constructor of the Window class.
- void setLoadingState ()
- void setGameState ()
- void setFailState ()
- void setWinState ()

- UI getUserInterface ()

Protected Attributes

- WindowState windowState

  Window's current state.
- TopBar topbar

  Window top bar.
- UI userInterface

  User interface which handle the game graphics inside the window.

7.37.1 Detailed Description

Definition at line 12 of file Window.java.

7.37.2 Constructor & Destructor Documentation

7.37.2.1 window.Window.Window ( )

Constructor of the Window class.

Definition at line 33 of file Window.java.

```
33                  {
34        Philophobia.getVerbose().calls("Creating Window class", "window/Window.java", "Window.Window()");
35
36        this.setTitle("Philophobia");
37        this.setSize(((int) Toolkit.getDefaultToolkit().getScreenSize().getWidth()), ((int) Toolkit.
      getDefaultToolkit().getScreenSize().getHeight()));
38        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
39
40        // Fullscreen mode
41        this.setExtendedState(this.getExtendedState() | JFrame.MAXIMIZED_BOTH);
42        this.setUndecorated(true);
43
44        this.setLayout(new BorderLayout());
45
46        windowState = WindowState.FirstLoading;
47
48        userInterface = new UI(this.getHeight(), this.getWidth());
49 //     this.getContentPane().add(userInterface, BorderLayout.CENTER);
50
51        topbar = new TopBar();
52        this.getContentPane().add(topbar, BorderLayout.PAGE_START);
53        this.pack();
54        this.setVisible(true);
55
56    }
```

7.37.3 Member Function Documentation

7.37.3.1 UI window.Window.getUserInterface ( )

Definition at line 83 of file Window.java.

```
83                               {
84        return userInterface;
85    }
```

7.37.3.2 void window.Window.setFailState ( )

Definition at line 71 of file Window.java.

```
                                {
        Philophobia.getVerbose().information("Setting window's Fail mode", "window/Window.java", "
    Window.setFailState()");
        topbar.hidePauseButton();
        windowState = WindowState.Fail;
    }
```

7.37.3.3 void window.Window.setGameState ( )

Definition at line 64 of file Window.java.

```
                                {
        Philophobia.getVerbose().information("Setting window's Game mode", "window/Window.java", "
    Window.setGameState()");
        topbar.showPauseButton();
        this.getContentPane().add(userInterface, BorderLayout.CENTER);
        windowState = WindowState.Game;
    }
```

7.37.3.4 void window.Window.setLoadingState ( )

Definition at line 58 of file Window.java.

```
                                   {
        Philophobia.getVerbose().information("Setting window's Loading mode", "window/Window.java", "
    Window.setLoadingState()");
        topbar.hidePauseButton();
        windowState = WindowState.LoadingScreen;
    }
```

7.37.3.5 void window.Window.setWinState ( )

Definition at line 77 of file Window.java.

```
                               {
        Philophobia.getVerbose().information("Setting window's Win mode", "window/Window.java", "
    Window.setWindState()");
        topbar.hidePauseButton();
        windowState = WindowState.Win;
    }
```

7.37.4 Member Data Documentation

7.37.4.1 TopBar window.Window.topbar [protected]

Window top bar.

Definition at line 23 of file Window.java.

7.37.4.2 UI window.Window.userInterface [protected]

User interface which handle the game graphics inside the window.

Definition at line 28 of file Window.java.

7.37.4.3 WindowState window.Window.windowState `[protected]`

Window's current state.

See Also

window.WindowState

Definition at line 18 of file Window.java.

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

- window/Window.java

7.38 window.WindowState Enum Reference

Enumeration of the possible states of the displayed graphics.

Public Attributes

- FirstLoading

  The FirstLoading state correspond to the state where the window has just been called and a special animation is displayed.
- Game

  The Game state correspond to the state in which the player is playing the game.
- LoadingScreen

  The LoadingScreen state correspond to the state where a loading animation is displayed.
- Win

  The Win state correspond to the state where the player has won the game and a congratulation screen is displayed.
- Fail

  The Fail state correspond to the state where the player has lost the game and a condolence screen is displayed.

7.38.1 Detailed Description

Enumeration of the possible states of the displayed graphics.

The state can be the "in game" state, a loading screen, the "win" or "fail" screen

See Also

Window
FirstLoading
Game
LoadingScreen
Win
Fail

Definition at line 101 of file Window.java.

7.38.2 Member Data Documentation

7.38.2.1 window.WindowState.Fail

The Fail state correspond to the state where the player has lost the game and a condolence screen is displayed.

Definition at line 136 of file Window.java.

7.38.2.2 window.WindowState.FirstLoading

The FirstLoading state correspond to the state where the window has just been called and a special animation is displayed.

Definition at line 109 of file Window.java.

7.38.2.3 window.WindowState.Game

The Game state correspond to the state in which the player is playing the game.

Definition at line 115 of file Window.java.

7.38.2.4 window.WindowState.LoadingScreen

The LoadingScreen state correspond to the state where a loading animation is displayed.

Definition at line 122 of file Window.java.

7.38.2.5 window.WindowState.Win

The Win state correspond to the state where the player has won the game and a congratulation screen is displayed.

Definition at line 129 of file Window.java.

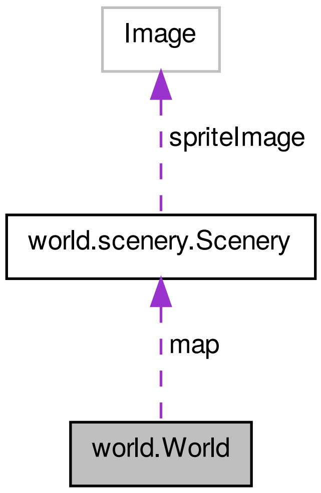
The documentation for this enum was generated from the following file:

- window/Window.java

7.39 world.World Class Reference

World is a class used to handle the game world graphics.

Collaboration diagram for world.World:



Public Member Functions

- World (final String type)

  World class constructor.
- World (final String type, final int sizeX, final int sizeY)

  World class constructor with size parameters.
- int getSizeX ()

  Getter for the sizeX Field.
- int getSizeY ()

  Getter for the sizeY Field.
- Scenery[ ][ ] getMap ()

  Getter for the map Field.

Protected Member Functions

- void generateWorld ()

  Function randomly generating the world.

Protected Attributes

- int sizeX

  Horizontal size of the map.
- int sizeY

  Vertical size of the map.
- Scenery[ ][ ] map

  Two-dimensional array containing all objects within the world.
- String type

  Style of the world.

7.39.1 Detailed Description

World is a class used to handle the game world graphics.

The World class use the Scenery objects (or the objects implementing the Scenery class) to display images of several objects corresponding to trees and traps and so on

Definition at line 18 of file World.java.

7.39.2 Constructor & Destructor Documentation

7.39.2.1 world.World.World (final String *type*)

World class constructor.

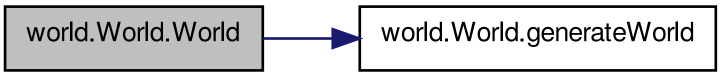
Parameters

type	Style of the world

Definition at line 47 of file World.java.

```
                                  {
        Philophobia.getVerbose().information("Creating World class", "world/World.java", "
    World.World(String)");

        this.type = type;
        this.sizeX = 128;
        this.sizeY = 128;

        generateWorld();
    }
```

Here is the call graph for this function:



7.39.2.2 world.World.World (final String *type*, final int *sizeX*, final int *sizeY*)

World class constructor with size parameters.

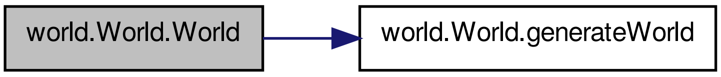
Parameters

type	Style of the world
sizeX	Horizontal size of the world
sizeY	Vertical size of the world

Definition at line 63 of file World.java.

```
                                                                        {
        Philophobia.getVerbose().information("Creating World class", "world/World.java", "
    World.World(String, int, int)");

        this.type = type;
        this.sizeX = sizeX;
        this.sizeY = sizeY;

        generateWorld();
    }
```

Here is the call graph for this function:



7.39.3 Member Function Documentation

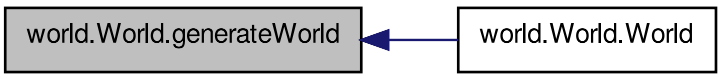
7.39.3.1 void world.World.generateWorld ( ) [protected]

Function randomly generating the world.

Definition at line 76 of file World.java.

```
                                {
        map = new Scenery[sizeX][sizeY];

        // Loop generating the world
        for(int i = 0 ; i < sizeX ; ++i) {
            for(int j = 0 ; j < sizeY ; ++j) {

                double random = Math.random();

                Scenery currentScenery;

                if(0 <= random && random < .3) {
                    currentScenery = new Tree(type);
                } else if(.3 <= random && random < .6) {
                    currentScenery = new Rock(type);
                } else {
                    currentScenery = new Ground(type);
                }

                map[i][j] = currentScenery;
            }
        }
    }
```

Here is the caller graph for this function:



7.39.3.2 Scenery [][] world.World.getMap ( )

Getter for the map Field.

Returns

Scenery[][] The map of the world

See Also

map

Definition at line 123 of file World.java.

```
                                {
        return this.map;
    }
```

7.39.3.3 int world.World.getSizeX ( )

Getter for the sizeX Field.

Returns

int Horizontal size of the map (in Scenery objects)

See Also

sizeX

Definition at line 105 of file World.java.

```
                              {
            return this.sizeX;
       }
```

7.39.3.4 int world.World.getSizeY ( )

Getter for the sizeY Field.

Returns

int Vertical size of the map (in Scenery objects)

See Also

sizeY

Definition at line 114 of file World.java.

```
                              {
            return this.sizeY;
       }
```

7.39.4 Member Data Documentation

7.39.4.1 Scenery [ ][ ] world.World.map [protected]

Two-dimensional array containing all objects within the world.

This object is filled using a file in the root directory and in the form of map[x][y] from top-left to right-bottom

Definition at line 36 of file World.java.

7.39.4.2 int world.World.sizeX [protected]

Horizontal size of the map.

Definition at line 23 of file World.java.

7.39.4.3 int world.World.sizeY [protected]

Vertical size of the map.

Definition at line 28 of file World.java.

7.39.4.4 String world.World.type [protected]

Style of the world.

Definition at line 41 of file World.java.

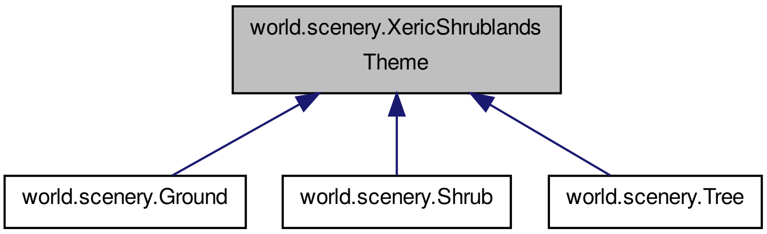
The documentation for this class was generated from the following file:

- world/World.java

7.40 world.scenery.XericShrublandsTheme Interface Reference

Interface used to define that a Scenery object have a Xeric Shrublands style.

Inheritance diagram for world.scenery.XericShrublandsTheme:



Static Public Attributes

- static String XERIC_SHRUBLANDS = "xericshrublands"

  String used to tell in which file is the sprite matching the Xeric Shrublands style.

7.40.1 Detailed Description

Interface used to define that a Scenery object have a Xeric Shrublands style.

Definition at line 7 of file XericShrublandsTheme.java.

7.40.2 Member Data Documentation

7.40.2.1 String world.scenery.XericShrublandsTheme.XERIC_SHRUBLANDS = "xericshrublands" `[static]`

String used to tell in which file is the sprite matching the Xeric Shrublands style.

Definition at line 13 of file XericShrublandsTheme.java.

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

- world/scenery/XericShrublandsTheme.java

Chapter 8

File Documentation

8.1 debug/CliOptions.java File Reference

Classes

- class debug.CliOptions

  Class used to analyze the commands passed to the program.

Packages

- package debug

8.2 debug/Verbose.java File Reference

Classes

- class debug.Verbose

  Class used to display and log messages all over this program.

Packages

- package debug

8.3 gameplay/ai/AI.java File Reference

Classes

- class gameplay.ai.AI

  Class representing the sadistic robot.

Packages

- package gameplay.ai

8.4 gameplay/ai/mood/Anger.java File Reference

Classes

- class gameplay.ai.mood.Anger

Packages

- package gameplay.ai.mood

8.5 gameplay/ai/mood/Curiosity.java File Reference

Classes

- class gameplay.ai.mood.Curiosity

Packages

- package gameplay.ai.mood

8.6 gameplay/ai/mood/Depression.java File Reference

Classes

- class gameplay.ai.mood.Depression

Packages

- package gameplay.ai.mood

8.7 gameplay/ai/mood/Mood.java File Reference

Classes

- class gameplay.ai.mood.Mood

  Abstract class representing an AI mood.
- class gameplay.ai.mood.ProbabilityMood

  Class used to associate a mood with a probability.

Packages

- package gameplay.ai.mood

8.8 gameplay/ai/mood/PowerComplex.java File Reference

Classes

- class gameplay.ai.mood.PowerComplex

Packages

- package gameplay.ai.mood

8.9 gameplay/ai/phrasing/OrderedPhrases.java File Reference

Classes

- class gameplay.ai.phrasing.OrderedPhrases

  Class used to contain an ordered list of phrases.

Packages

- package gameplay.ai.phrasing

8.10 gameplay/ai/phrasing/Phrasing.java File Reference

Classes

- class gameplay.ai.phrasing.Phrasing

Packages

- package gameplay.ai.phrasing

8.11 gameplay/GamePlay.java File Reference

Classes

- class gameplay.GamePlay

  Class handling the game play of this game.

Packages

- package gameplay

8.12 main/Philophobia.java File Reference

Classes

- class main.Philophobia

  Main class.

Packages

- package main

8.13 README.md File Reference

8.14 window/topbar/TopBar.java File Reference

Classes

- class window.topbar.TopBar

    Class handling the top bar of the program's main window.

Packages

- package window.topbar

8.15 window/topbar/TopBarButton.java File Reference

Classes

- class window.topbar.TopBarButton

    Class handling the UI buttons.

Packages

- package window.topbar

8.16 window/ui/UI.java File Reference

Classes

- class window.ui.UI

    Class used to handle the window's user interface.

Packages

- package window.ui

8.17 window/Window.java File Reference

Classes

- class window.Window
- enum window.WindowState

    Enumeration of the possible states of the displayed graphics.

Packages

- package window

8.18 world/character/Character.java File Reference

Classes

- class world.character.Character

  Class used to handle a character (player or non-player)

Packages

- package world.character

8.19 world/character/Hero.java File Reference

Classes

- class world.character.Hero

Packages

- package world.character

8.20 world/character/TalkingCharacter.java File Reference

Classes

- class world.character.TalkingCharacter

  Class handling a talking character.

Packages

- package world.character

8.21 world/InteractiveObject.java File Reference

Classes

- interface world.InteractiveObject

Packages

- package world

8.22 world/scenery/AlpineTundraTheme.java File Reference

Classes

- interface world.scenery.AlpineTundraTheme

  Interface used to define that a Scenery object have an Alpine Tundra style.

Packages

- package world.scenery

8.23 world/scenery/CaveTheme.java File Reference

Classes

- interface world.scenery.CaveTheme

  Interface used to tell that a Scenery object has a Cave style.

Packages

- package world.scenery

8.24 world/scenery/Flower.java File Reference

Classes

- class world.scenery.Flower

  Class used to handle a flower object.

Packages

- package world.scenery

8.25 world/scenery/Grass.java File Reference

Classes

- class world.scenery.Grass

  Class used to handle a grass object.

Packages

- package world.scenery

8.26 world/scenery/Ground.java File Reference

Classes

- class world.scenery.Ground

  Class used to handle a ground object.

Packages

- package world.scenery

8.27 world/scenery/PolarDesertTheme.java File Reference

Classes

- interface world.scenery.PolarDesertTheme

  Interface used to define that a Scenery object has a Polar Desert style.

Packages

- package world.scenery

8.28 world/scenery/Rock.java File Reference

Classes

- class world.scenery.Rock

  Class used to handle a rock object.

Packages

- package world.scenery

8.29 world/scenery/SavannaTheme.java File Reference

Classes

- interface world.scenery.SavannaTheme

  Interface used to define that a Scenery object have a Savanna style.

Packages

- package world.scenery

8.30 world/scenery/Scenery.java File Reference

Classes

- class world.scenery.Scenery

  Class used to handle any world object.

Packages

- package world.scenery

8.31 world/scenery/Shore.java File Reference

Classes

- class world.scenery.Shore

  Class used to handle a shore object.

Packages

- package world.scenery

8.32 world/scenery/Shrub.java File Reference

Classes

- class world.scenery.Shrub

  Class used to handle a shrub object.

Packages

- package world.scenery

8.33 world/scenery/SteppeTheme.java File Reference

Classes

- interface world.scenery.SteppeTheme

  Interface used to define that a Scenery object have a Steppe style.

Packages

- package world.scenery

8.34 world/scenery/TaigaTheme.java File Reference

Classes

- interface world.scenery.TaigaTheme

  Interface used to define that a Scenery object have a Taiga style.

Packages

- package world.scenery

8.35 world/scenery/TemperateBroadleafTheme.java File Reference

Classes

- interface world.scenery.TemperateBroadleafTheme

  Interface used to define that a Scenery object have a Temperate Broadleaf style.

Packages

- package world.scenery

8.36 world/scenery/Tree.java File Reference

Classes

- class world.scenery.Tree

  Class used to handle a tree object.

Packages

- package world.scenery

8.37 world/scenery/Water.java File Reference

Classes

- class world.scenery.Water

  Class used to handle a water object.

Packages

- package world.scenery

8.38 world/scenery/XericShrublandsTheme.java File Reference

Classes

- interface world.scenery.XericShrublandsTheme

  Interface used to define that a Scenery object have a Xeric Shrublands style.

Packages

- package world.scenery

8.39 world/World.java File Reference

Classes

- class world.World

  World is a class used to handle the game world graphics.

Packages

- package world

Index