Plane Simulator

Généré par Doxygen 1.8.13

Table des matières

Chapitre 1

Index hiérarchique

1.1 Hiérarchie des classes

Cette liste d'héritage est classée approximativement par ordre alphabétique :

Chapitre 2

Index des classes

2.1 Liste des classes

Liste des classes, structures, unions et interfaces avec une brève description :

Chapitre 3

Index des fichiers

3.1 Liste des fichiers

Liste de tous les fichiers avec une brève description :

Chapitre 4

Documentation des classes

4.1 Référence de la classe App

`#include <App.h>`

Fonctions membres publiques

- void initWindow ()
  Initialisation de la fenêtre de connexion.
- App ()
  Construct a new App : : App object.
- void setState (State *newState)
  Actuellement connecté
- void run ()
  Lancement de l'application.
- void init (std : :string id)
  Initialise un historique.
- void printHistory ()
  Affiche l'historique de l'utilisateur connecté
- void checkRating ()
  Vérifie la note.
- void zero ()
  Attribue un zéro.
- void addRating (int rate)
  Ajoute une note.
- void addDate ()
  Met à jour la date.
- void addHistory (std : :string str)
  Met à jour l'historique.
- void printJson ()
  Crée le Json.

4.1.1 Description détaillée

Définition à la ligne 14 du fichier App.h.

4.1.2 Documentation des constructeurs et destructeur

4.1.2.1 App()

```
App::App ( )
```

Construct a new App : : App object.

Définition à la ligne 18 du fichier App.cpp.

```
18          : currentState(new Login)
19 {
20     // this->rate = 0;
21 }
```

4.1.3 Documentation des fonctions membres

4.1.3.1 addDate()

```
void App::addDate ( )
```

Met à jour la date.

Définition à la ligne 130 du fichier App.cpp.

```
131 {
132     time_t rawtime;
133     struct tm *timeinfo;
134     char buffer[80];
135
136     time(&rawtime);
137     timeinfo = localtime(&rawtime);
138
139     strftime(buffer, sizeof(buffer), "%d-%m-%Y %H:%M:%S", timeinfo);
140     std::string str(buffer);
141     this->j[this->m_id]["date"][this->number] = str;
142 }
```

4.1.3.2 addHistory()

```
void App::addHistory (
             std::string str )
```

Met à jour l'historique.

Paramètres

str	

Définition à la ligne 149 du fichier App.cpp.

```
150 {
151     int size = this->j[this->m_id]["history"][this->number].size();
152     this->j[this->m_id]["history"][this->number][size] = str;
153 }
```

4.1.3.3 addRating()

```
void App::addRating (
            int rate )
```

Ajoute une note.

Paramètres

rate	

Définition à la ligne 98 du fichier App.cpp.

```
99 {
100     this->rate = this->rate + rate;
101 }
```

4.1.3.4 checkRating()

```
void App::checkRating ( )
```

Vérifie la note.

Définition à la ligne 107 du fichier App.cpp.

```
108 {
109     int size = this->j[this->m_id]["rating"].size();
110     // cond qui test si on a deja ajouter des point
111     if (size == this->number)
112     {
113         this->rate = 0;
114     }
115 }
```

4.1.3.5 init()

```
void App::init (
            std::string id )
```

Initialise un historique.

Définition à la ligne 37 du fichier App.cpp.

Référencé par Login : :run().

```
38 {
39     std::ifstream i("file.json");
40     i >> this->j;
41
42     this->m_id = id;
43
44     this->number = this->j[this->m_id]["date"].size();
45 }
```

Voici le graphe des appelants de cette fonction :

4.1.3.6 initWindow()

```
void App::initWindow ( )
```

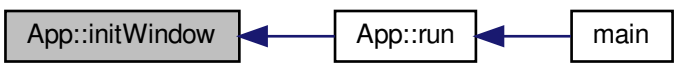
Initialisation de la fenêtre de connexion.

Définition à la ligne 9 du fichier App.cpp.

Référencé par run().

```
10 {
11     InitWindow(400, 600, "App");
12 }
```

Voici le graphe des appelants de cette fonction :



4.1.3.7 printHistory()

```
void App::printHistory ( )
```

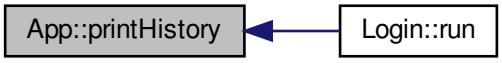
Affiche l'historique de l'utilisateur connecté

Définition à la ligne 51 du fichier App.cpp.

Référencé par Login : :run().

```
52 {
53     time_t rawtime;
54     struct tm *timeinfo;
55     char buffer[80];
56 
57     time(&rawtime);
58     timeinfo = localtime(&rawtime);
59 
60     strftime(buffer, sizeof(buffer), "%d-%m-%Y %H:%M:%S", timeinfo);
61     std::string str(buffer);
62     std::cout << std::endl;
63     std::cout << "We are the " << str << std::endl;
64     std::cout << "History :" << std::endl;
65 
66     int size = this->number;
67 
68     if (size == 0)
69     {
70         std::cout << "No history" << std::endl;
71         return;
72     }
73 
74     while (size > 0)
75     {
76         size--;
77         std::cout << "Date : " << this->j[this->m_id]["date"][size] << std::endl;
78         std::cout << "Rating : " << this->j[this->m_id]["rating"][size] << std::endl;
79 
80         std::cout << "App : " << std::endl;
81         int historySize = this->j[this->m_id]["history"][size].size();
82         int i = 0;
83         while (i < historySize)
84         {
85             std::cout << "\t" << i + 1 << " - " << this->j[this->m_id]["history"][size][i] << std::endl;
86             i++;
87         }
88         std::cout << std::endl
89                   << std::endl;
90     }
91 }
```

Voici le graphe des appelants de cette fonction :



4.1.3.8 printJson()

```
void App::printJson ( )
```

Crée le Json.

Définition à la ligne 159 du fichier App.cpp.

```
160 {
161     this->j[this->m_id]["rating"][this->number] = this->rate;
162     std::ofstream o("file.json");
163     o << this->j;
164 }
```

4.1.3.9 run()

```
void App::run ( )
```

Lancement de l'application.

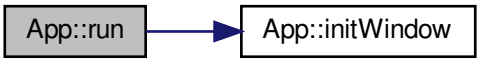
Définition à la ligne 170 du fichier App.cpp.

Références initWindow().

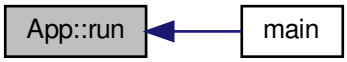
Référencé par main().

```
171 {
172     this->initWindow();
173     SetTargetFPS(60);
174 
175     while (!WindowShouldClose())
176     {
177         BeginDrawing();
178 
179         ClearBackground(WHITE);
180         currentState->run(this);
181         EndDrawing();
182     }
183 
184     CloseWindow();
185 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.1.3.10 setState()

```
void App::setState (
            State * newState )
```

Actuellement connecté

Paramètres

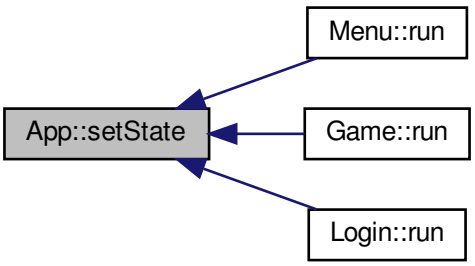
newState	

Définition à la ligne 28 du fichier App.cpp.

Référencé par Menu : :run(), Game : :run(), et Login : :run().

```
29 {
30     currentState.reset(newState);
31 }
```

Voici le graphe des appelants de cette fonction :



4.1.3.11 zero()

```
void App::zero ( )
```

Attribue un zéro.

Définition à la ligne 121 du fichier App.cpp.

```
122 {
123     this->rate = 0;
124 }
```

La documentation de cette classe a été générée à partir des fichiers suivants :

- include/App.h
- src/App.cpp

4.2 Référence de la classe DashBoard

```
#include <DashBoard.h>
```

Fonctions membres publiques

- DashBoard ()
  Construct a new Dash Board : : Dash Board object.
- void createButton ()
  Liste des boutons du dash board.
- void createButtonName ()
  Nom des boutons.
- void drawDashBoard ()
  Dessine les composants graphiques.
- void mouseOn ()
  Détection de la position de la souris.
- void clicOn (FuelSystem *)
  Pression sur un bouton.

4.2.1 Description détaillée

Définition à la ligne 7 du fichier DashBoard.h.

4.2.2 Documentation des constructeurs et destructeur

4.2.2.1 DashBoard()

```
DashBoard::DashBoard ( )
```

Construct a new Dash Board : : Dash Board object.

Définition à la ligne 8 du fichier DashBoard.cpp.

Références createButton().

```
8                       : FL_WIDTH(800), FL_HEIGHT(700)
9 {
10      createButton();
11 };
```

Voici le graphe d'appel pour cette fonction :

4.2.3 Documentation des fonctions membres

4.2.3.1 clicOn()

```
void DashBoard::clicOn (
             FuelSystem * fs )
```

Pression sur un bouton.

Paramètres

fs	

Définition à la ligne 111 du fichier DashBoard.cpp.

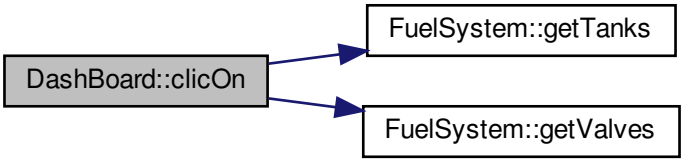
Références FuelSystem : :getTanks(), et FuelSystem : :getValves().

Référencé par Game : :run().

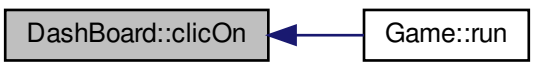
```
112 {
113     std::vector<Valve *> v_Valves = fs->getValves();
114     std::vector<Pump *> p_Pumps = {&fs->getTanks().at(0)->getEmergencyPump(), &fs->
      getTanks().at(1)->getEmergencyPump(), &fs->getTanks().at(2)->getEmergencyPump()};
```

```
115
116     //V12
117     if (CheckCollisionPointRec(GetMousePosition(), this->m_button.at(0)) && IsMouseButtonReleased(
      MOUSE_LEFT_BUTTON))
118         v_Valves.at(0)->update();
119     //V13
120     else if (CheckCollisionPointRec(GetMousePosition(), this->m_button.at(1)) && IsMouseButtonReleased(
      MOUSE_LEFT_BUTTON))
121         v_Valves.at(1)->update();
122     //V23
123     else if (CheckCollisionPointRec(GetMousePosition(), this->m_button.at(2)) && IsMouseButtonReleased(
      MOUSE_LEFT_BUTTON))
124         v_Valves.at(2)->update();
125     //VT12
126     else if (CheckCollisionPointRec(GetMousePosition(), this->m_button.at(3)) && IsMouseButtonReleased(
      MOUSE_LEFT_BUTTON))
127         v_Valves.at(3)->update();
128     //VT23
129     else if (CheckCollisionPointRec(GetMousePosition(), this->m_button.at(4)) && IsMouseButtonReleased(
      MOUSE_LEFT_BUTTON))
130         v_Valves.at(4)->update();
131
132     //P12
133     else if (CheckCollisionPointRec(GetMousePosition(), this->m_button.at(5)) && IsMouseButtonReleased(
      MOUSE_LEFT_BUTTON))
134         p_Pumps.at(0)->update();
135
136     //P22
137     else if (CheckCollisionPointRec(GetMousePosition(), this->m_button.at(6)) && IsMouseButtonReleased(
      MOUSE_LEFT_BUTTON))
138         p_Pumps.at(1)->update();
139
140     //P32
141     else if (CheckCollisionPointRec(GetMousePosition(), this->m_button.at(7)) && IsMouseButtonReleased(
      MOUSE_LEFT_BUTTON))
142         p_Pumps.at(2)->update();
143 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.2.3.2 createButton()

```
void DashBoard::createButton ( )
```

Liste des boutons du dash board.

Définition à la ligne 17 du fichier DashBoard.cpp.

Références FL_HEIGHT, et FL_WIDTH.

Référencé par DashBoard().

```
18 {
19     //V12
20     Rectangle r1 = {
21         (FL_WIDTH / 3),
22         (FL_HEIGHT / (float)1.1),
23         (FL_WIDTH / 10),
24         (FL_HEIGHT / 20),
25     };
26     m_button.push_back(r1);
27 
28     //V13
29     Rectangle r2 = {((FL_WIDTH / 2) - ((FL_WIDTH / 10) / 2)), (
       FL_HEIGHT / (float)1.1), (FL_WIDTH / 10), (FL_HEIGHT / 20)};
30     m_button.push_back(r2);
31 
32     //V23
33     Rectangle r3 = {(FL_WIDTH - ((FL_WIDTH / 3) + (FL_WIDTH / 10))), (
       FL_HEIGHT / (float)1.1), (FL_WIDTH / 10), (FL_HEIGHT / 20)};
34     m_button.push_back(r3);
35 
36     //VT12
37     Rectangle r4 = {(FL_WIDTH / (float)2.6), (FL_HEIGHT / (float)1.32), (
       FL_WIDTH / 10), (FL_HEIGHT / 20)};
38     m_button.push_back(r4);
39 
40     //VT23
41     Rectangle r5 = {(FL_WIDTH - ((FL_WIDTH / (float)2.6) + (
       FL_WIDTH / 10))), (FL_HEIGHT / (float)1.32), (FL_WIDTH / 10), (
       FL_HEIGHT / 20)};
42     m_button.push_back(r5);
43 
44     //P12
45     Rectangle r6 = {(FL_WIDTH / 3), (FL_HEIGHT / (float)(float)1.2), (
       FL_WIDTH / 10), (FL_HEIGHT / 20)};
46     m_button.push_back(r6);
47 
48     //P22
49     Rectangle r7 = {((FL_WIDTH / 2) - ((FL_WIDTH / 10) / 2)), (
       FL_HEIGHT / (float)1.2), (FL_WIDTH / 10), (FL_HEIGHT / 20)};
50     m_button.push_back(r7);
51 
52     //P32
53     Rectangle r8 = {(FL_WIDTH - ((FL_WIDTH / 3) + (FL_WIDTH / 10))), (
       FL_HEIGHT / (float)1.2), (FL_WIDTH / 10), (FL_HEIGHT / 20)};
54     m_button.push_back(r8);
55 }
```

Voici le graphe des appelants de cette fonction :

4.2.3.3 createButtonName()

```
void DashBoard::createButtonName ( )
```

Nom des boutons.

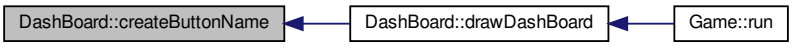
Définition à la ligne 61 du fichier DashBoard.cpp.

Références FL_WIDTH.

Référencé par drawDashBoard().

```
62 {
63     DrawText("V12", m_button.at(0).x + (m_button.at(0).width * 0.25), m_button.at(0).y + (m_button.at(0).
      height * 0.20), FL_WIDTH * 0.030, BLUE);
64     DrawText("V13", m_button.at(1).x + (m_button.at(1).width * 0.25), m_button.at(1).y + (m_button.at(1).
      height * 0.20), FL_WIDTH * 0.030, BLUE);
65     DrawText("V23", m_button.at(2).x + (m_button.at(2).width * 0.25), m_button.at(2).y + (m_button.at(2).
      height * 0.20), FL_WIDTH * 0.030, BLUE);
66     DrawText("VT12", m_button.at(3).x + (m_button.at(3).width * 0.15), m_button.at(3).y + (m_button.at(3).
      height * 0.20), FL_WIDTH * 0.030, BLUE);
67     DrawText("VT23", m_button.at(4).x + (m_button.at(4).width * 0.11), m_button.at(4).y + (m_button.at(4).
      height * 0.20), FL_WIDTH * 0.030, BLUE);
68     DrawText("P12", m_button.at(5).x + (m_button.at(5).width * 0.25), m_button.at(5).y + (m_button.at(5).
      height * 0.20), FL_WIDTH * 0.030, BLUE);
69     DrawText("P22", m_button.at(6).x + (m_button.at(6).width * 0.25), m_button.at(6).y + (m_button.at(6).
      height * 0.20), FL_WIDTH * 0.030, BLUE);
70     DrawText("P32", m_button.at(7).x + (m_button.at(7).width * 0.25), m_button.at(7).y + (m_button.at(7).
      height * 0.20), FL_WIDTH * 0.030, BLUE);
71 }
```

Voici le graphe des appelants de cette fonction :



4.2.3.4 drawDashBoard()

```
void DashBoard::drawDashBoard ( )
```

Dessine les composants graphiques.

Définition à la ligne 77 du fichier DashBoard.cpp.

Références createButtonName().

Référencé par Game : :run().

```
78 {
79     for (Rectangle r : m_button)
80     {
81         DrawRectangleRounded(r, 0.5, 4, LIGHTGRAY);
82     }
83     createButtonName();
84 }
```

Voici le graphe d'appel pour cette fonction :

Voici le graphe des appelants de cette fonction :

4.2.3.5 mouseOn()

```
void DashBoard::mouseOn ( )
```

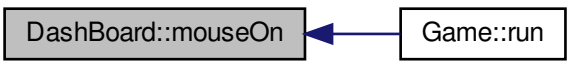
Détection de la position de la souris.

Définition à la ligne 90 du fichier DashBoard.cpp.

Référencé par Game : :run().

```
91 {
92     for (Rectangle r : m_button)
93     {
94         if (CheckCollisionPointRec(GetMousePosition(), r))
95         {
96             DrawRectangleRoundedLines(r, 0.5, 4, 5, BLUE);
97         }
98
99         else
100         {
101             DrawRectangleRoundedLines(r, 0.5, 4, 5, GRAY);
102         }
103     }
104 }
```

Voici le graphe des appelants de cette fonction :



La documentation de cette classe a été générée à partir des fichiers suivants :

- include/DashBoard.h
- src/DashBoard.cpp

4.3 Référence de la classe FuelSystem

`#include <FuelSystem.h>`

Fonctions membres publiques

- FuelSystem ()
  Construct a new Fuel System : : Fuel System object.
- std : :vector< Tank * > getTanks ()
  Représentation des Réservoirs.
- std : :vector< Motor * > getMotors ()
  Représentation des Moteurs.
- std : :vector< Valve * > getValves ()
  Représentation des Vannes.
- std : :vector< Pipe * > getPipe ()
  Représentation des Tubes.
- void initializeValveMap ()
  Connexion entre les Moteurs et les Réservoirs.
- bool allTanksEmpty ()
  Retourne si les réservoirs sont vides ou pas.
- void run ()
  Tubes opérationnels.
- void drawPipe ()
  Dessine les tubes.
- void drawFuelSystem ()
  Dessine le schéma complet.

4.3.1 Description détaillée

Définition à la ligne 17 du fichier FuelSystem.h.

4.3.2 Documentation des constructeurs et destructeur

4.3.2.1 FuelSystem()

`FuelSystem::FuelSystem ( )`

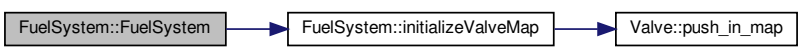
Construct a new Fuel System : : Fuel System object.

Définition à la ligne 7 du fichier FuelSystem.cpp.

Références initializeValveMap().

```
7                         : m_T1("Tank1", 200, "P11", "P12"), m_T2("Tank2", 100, "P21", "P22"), m_T3("Tank3",
      200, "P31", "P32"),
8                           m_M1("M1", &m_T1.getPrimaryPump()), m_M2("M2", &m_T2.
      getPrimaryPump()), m_M3("M3", &m_T3.getPrimaryPump()),
9                           m_V12("V12"), m_V13("V13"), m_V23("V23"), m_VT12("VT12"), m_VT23("VT23"),
10                           m_T1_to_T2(&m_T1, &m_VT12, &m_T2), m_T2_to_T3(&m_T2, &m_VT23, &m_T3),
11                           m_P_T1(&m_T1, &m_V12, &m_V13, &m_M1), m_P_T2(&m_T2, &m_V12, &m_V23, &m_M2),
      m_P_T3(&m_T3, &m_V13, &m_V23, &m_M3)
12 {
13     initializeValveMap();
14 }
```

Voici le graphe d'appel pour cette fonction :



4.3.3 Documentation des fonctions membres

4.3.3.1 allTanksEmpty()

```
bool FuelSystem::allTanksEmpty ( )
```

Retourne si les réservoirs sont vides ou pas.

Renvoie

true
false

Définition à la ligne 82 du fichier FuelSystem.cpp.

Références Tank : :isEmpty().

Référencé par Game : :run().

```
83 {
84     if (m_T1.isEmpty() && m_T2.isEmpty() && m_T3.isEmpty())
85     {
86         return true;
87     }
88
89     return false;
90 }
```

Voici le graphe d'appel pour cette fonction :

Voici le graphe des appelants de cette fonction :

4.3.3.2 drawFuelSystem()

```
void FuelSystem::drawFuelSystem ( )
```

Dessine le schéma complet.

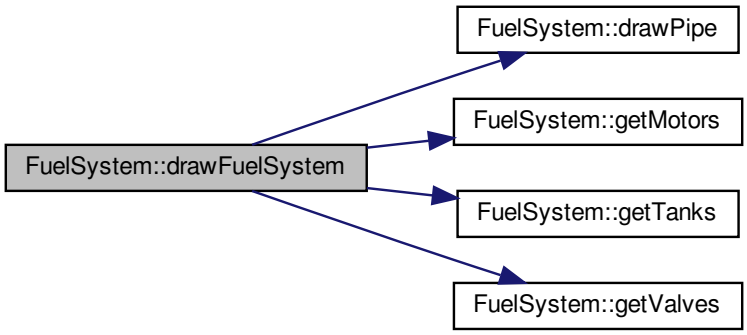
Définition à la ligne 116 du fichier FuelSystem.cpp.

Références drawPipe(), getMotors(), getTanks(), et getValves().

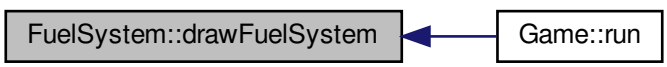
Référencé par Game : :run().

```
117 {
118     drawPipe();
119
120     std::vector<Tank *> v_tank = getTanks();
121     std::vector<Motor *> v_Motors = getMotors();
122     std::vector<Valve *> v_Valves = getValves();
123
124     for (auto t : v_tank)
125     {
126         t->run();
127     }
128
129     for (Motor *m : v_Motors)
130     {
131         m->drawMotor();
132     }
133
134     for (Valve *v : v_Valves)
135     {
136         v->drawValve();
137     }
138 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.3.3.3 drawPipe()

```
void FuelSystem::drawPipe ( )
```

Dessine les tubes.

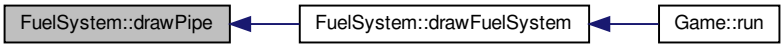
Définition à la ligne 96 du fichier FuelSystem.cpp.

Références FL_HEIGHT, et FL_WIDTH.

Référencé par drawFuelSystem().

```
97 {
98     DrawLine((FL_WIDTH / 8) + (FL_WIDTH / 6), (FL_HEIGHT / 50) + (
      FL_HEIGHT / 5) / 2, ((FL_WIDTH / 2) - ((FL_WIDTH / 8) / 2)), (
      FL_HEIGHT / 50) + (FL_HEIGHT / 5) / 2, BLACK);
99     DrawLine(((FL_WIDTH / 2) - ((FL_WIDTH / 8) / 2)) + (FL_WIDTH / 8), (
      FL_HEIGHT / 50) + (FL_HEIGHT / 5) / 2, (FL_WIDTH - ((
      FL_WIDTH / 8) + (FL_WIDTH / 6))), (FL_HEIGHT / 50) + (
      FL_HEIGHT / 5) / 2, BLACK);
100     DrawLine((FL_WIDTH / 8) + ((FL_WIDTH / 6) / 2), (FL_HEIGHT - (
      FL_HEIGHT / 2.5)) * 0.7, (FL_WIDTH / 8) + ((FL_WIDTH / 6) / 2), (
      FL_HEIGHT - (FL_HEIGHT / 2.5)), BLACK);
101     DrawLine((FL_WIDTH / 8) + ((FL_WIDTH / 6) / 2), (FL_HEIGHT - (
      FL_HEIGHT / 2.5)) * 0.7, (FL_WIDTH / 8) + ((FL_WIDTH / 6) / 1.5), (
      FL_HEIGHT - (FL_HEIGHT / 2.5)) * 0.7, BLACK);
102     DrawLine((FL_WIDTH / 8) + ((FL_WIDTH / 6) / 1.5), (FL_HEIGHT / 50) + (
      FL_HEIGHT / 5), (FL_WIDTH / 8) + ((FL_WIDTH / 6) / 1.5), (
      FL_HEIGHT - (FL_HEIGHT / 2.5)) * 0.8, BLACK);
103     DrawLine(((FL_WIDTH / 2) - ((FL_WIDTH / 8) / 2)) + ((
      FL_WIDTH / 8) / 2), (FL_HEIGHT / 50) + (FL_HEIGHT / 5), (((
      FL_WIDTH / 2) - ((FL_WIDTH / 8) / 2)) + (FL_WIDTH / 8) / 2 - (
      FL_WIDTH / 14) / 2) + ((FL_WIDTH / 14) / 2), (FL_HEIGHT - (
      FL_HEIGHT / 2.5)), BLACK);
104     DrawLine((FL_WIDTH / 8) + ((FL_WIDTH / 6) / 1.5), (FL_HEIGHT - (
      FL_HEIGHT / 2.5)) * 0.8, ((FL_WIDTH / 2) - ((FL_WIDTH / 8) / 2)) + ((
      FL_WIDTH / 8) / 2), (FL_HEIGHT - (FL_HEIGHT / 2.5)) * 0.8, BLACK);
105     DrawLine((FL_WIDTH / 8) + ((FL_WIDTH / 6) / 1.5), (FL_HEIGHT - (
      FL_HEIGHT / 2.5)) * 0.6, (FL_WIDTH - ((FL_WIDTH / 8) + (
      FL_WIDTH / 6))) + ((FL_WIDTH / 6) / 2), (FL_HEIGHT - (
      FL_HEIGHT / 2.5)) * 0.6, BLACK);
106     DrawLine((FL_WIDTH - ((FL_WIDTH / 8) + (FL_WIDTH / 6))) + ((
      FL_WIDTH / 6) / 2), (FL_HEIGHT - (FL_HEIGHT / 2.5)) * 0.6, (
      FL_WIDTH - ((FL_WIDTH / 8) + (FL_WIDTH / 6))) + ((
      FL_WIDTH / 6) / 2), (FL_HEIGHT - (FL_HEIGHT / 2.5)), BLACK);
107     DrawLine((FL_WIDTH - ((FL_WIDTH / 8) + (FL_WIDTH / 6))) + ((
      FL_WIDTH / 6) / 3), (FL_HEIGHT / 50) + (FL_HEIGHT / 5), (
      FL_WIDTH - ((FL_WIDTH / 8) + (FL_WIDTH / 6))) + ((
      FL_WIDTH / 6) / 3), (FL_HEIGHT - (FL_HEIGHT / 2.5)) * 0.9, BLACK);
108     DrawLine((FL_WIDTH - ((FL_WIDTH / 8) + (FL_WIDTH / 6))) + ((
      FL_WIDTH / 6) / 3), (FL_HEIGHT - (FL_HEIGHT / 2.5)) * 0.9, ((
      FL_WIDTH / 2) - ((FL_WIDTH / 8) / 2)) + ((FL_WIDTH / 8) / 2), (
      FL_HEIGHT - (FL_HEIGHT / 2.5)) * 0.9, BLACK);
109     DrawLine((FL_WIDTH - ((FL_WIDTH / 8) + (FL_WIDTH / 6))) + ((
      FL_WIDTH / 6) / 3), (FL_HEIGHT - (FL_HEIGHT / 2.5)) * 0.7, (
      FL_WIDTH - ((FL_WIDTH / 8) + (FL_WIDTH / 6))) + ((
      FL_WIDTH / 6) / 2), (FL_HEIGHT - (FL_HEIGHT / 2.5)) * 0.7, BLACK);
110 }
```

Voici le graphe des appelants de cette fonction :



4.3.3.4 getMotors()

```
std::vector< Motor * > FuelSystem::getMotors ( )
```

Représentation des Moteurs.

Renvoie

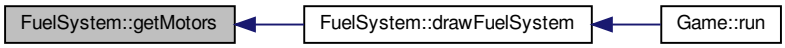
std : :vector<Motor *>

Définition à la ligne 32 du fichier FuelSystem.cpp.

Référencé par drawFuelSystem().

```
33 {
34     std::vector<Motor *> v_Motors = {&m_M1, &m_M2, &m_M3};
35     return v_Motors;
36 }
```

Voici le graphe des appelants de cette fonction :



4.3.3.5 getPipe()

```
std::vector< Pipe * > FuelSystem::getPipe ( )
```

Représentation des Tubes.

Renvoie

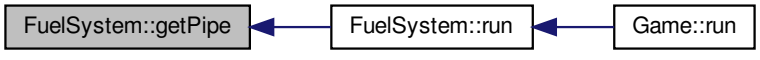
std : :vector<Pipe *>

Définition à la ligne 54 du fichier FuelSystem.cpp.

Référencé par run().

```
55 {
56     std::vector<Pipe *> v_Pipe = {&m_T1_to_T2, &m_T2_to_T3, &m_P_T1, &m_P_T2, &m_P_T3};
57     return v_Pipe;
58 }
```

Voici le graphe des appelants de cette fonction :



4.3.3.6 getTanks()

```
std::vector< Tank * > FuelSystem::getTanks ( )
```

Représentation des Réservoirs.

Renvoie

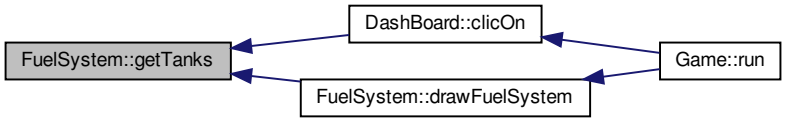
std : :vector<Tank *>

Définition à la ligne 21 du fichier FuelSystem.cpp.

Référencé par DashBoard : :clicOn(), et drawFuelSystem().

```
22 {
23     std::vector<Tank *> v_Tanks = {&m_T1, &m_T2, &m_T3};
24     return v_Tanks;
25 }
```

Voici le graphe des appelants de cette fonction :



4.3.3.7 getValves()

```
std::vector< Valve * > FuelSystem::getValves ( )
```

Représentation des Vannes.

Renvoie

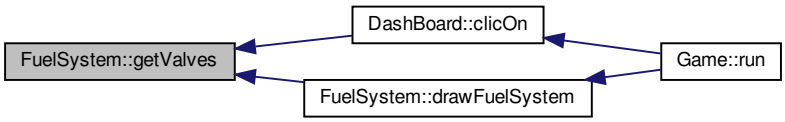
std : :vector<Valve *>

Définition à la ligne 43 du fichier FuelSystem.cpp.

Référencé par DashBoard : :clicOn(), et drawFuelSystem().

```
44 {
45     std::vector<Valve *> v_Valves = {&m_V12, &m_V13, &m_V23, &m_VT12, &m_VT23};
46     return v_Valves;
47 }
```

Voici le graphe des appelants de cette fonction :



4.3.3.8 initializeValveMap()

```
void FuelSystem::initializeValveMap ( )
```

Connexion entre les Moteurs et les Réservoirs.

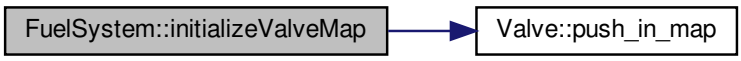
Définition à la ligne 64 du fichier FuelSystem.cpp.

Références Valve : :push_in_map().

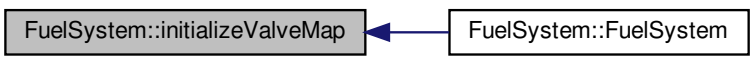
Référencé par FuelSystem().

```
65 {
66     m_V12.push_in_map(&m_T1, &m_M2);
67     m_V12.push_in_map(&m_T2, &m_M1);
68 
69     m_V13.push_in_map(&m_T1, &m_M3);
70     m_V13.push_in_map(&m_T3, &m_M1);
71 
72     m_V23.push_in_map(&m_T2, &m_M3);
73     m_V23.push_in_map(&m_T3, &m_M2);
74 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.3.3.9 run()

```
void FuelSystem::run ( )
```

Tubes opérationnels.

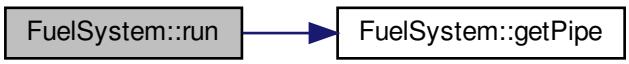
Définition à la ligne 144 du fichier FuelSystem.cpp.

Références getPipe().

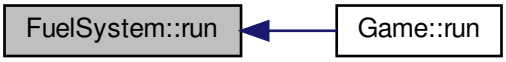
Référencé par Game : :run().

```
145 {
146     std::vector<Pipe *> v_Pipe = getPipe();
147 
148     for (Pipe *p : v_Pipe)
149     {
150         p->run();
151     }
152 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



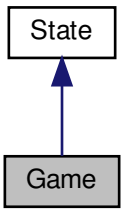
La documentation de cette classe a été générée à partir des fichiers suivants :

- include/FuelSystem.h
- src/FuelSystem.cpp

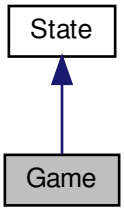
4.4 Référence de la classe Game

`#include <Game.h>`

Graphe d'héritage de Game :



Graphe de collaboration de Game :



Fonctions membres publiques

- Rectangle getBackBlock ()
  Retourne une représentation Graphique.
- void checkCollisionBlock ()
  Vérifie les cliques de la souris.
- void drawBackBlock ()
  Dessine un block.
- Game ()
  Construct a new Game : : Game object.
- void run (App ∗app)
  Lance l'exercice de simulation.
- void isBack (App ∗app)

Membres hérités additionnels

4.4.1 Description détaillée

Définition à la ligne 10 du fichier Game.h.

4.4.2 Documentation des constructeurs et destructeur

4.4.2.1 Game()

```
Game::Game ( )
```

Construct a new Game : : Game object.

Définition à la ligne 9 du fichier Game.cpp.

Références State : :W_HEIGHT, State : :W_TITLE, et State : :W_WIDTH.

```
9             : State(800, 700, (char *)"Game"), m_fs(), m_db()
10 {
11     SetWindowSize(W_WIDTH, W_HEIGHT);
12     SetWindowTitle(W_TITLE);
13 };
```

4.4.3 Documentation des fonctions membres

4.4.3.1 checkCollisionBlock()

```
void Game::checkCollisionBlock ( )
```

Vérifie les cliques de la souris.

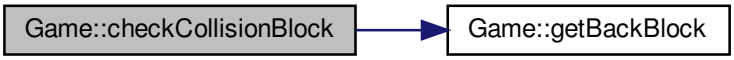
Définition à la ligne 30 du fichier Game.cpp.

Références getBackBlock().

Référencé par run().

```
31 {
32     if (CheckCollisionPointRec(GetMousePosition(), getBackBlock()))
33     {
34         mouseOnBack = true;
35 
36         if (IsMouseButtonReleased(MOUSE_LEFT_BUTTON))
37         {
38             clicOnBack = true;
39         }
40     }
41     else
42         mouseOnBack = false;
43 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :

4.4.3.2 drawBackBlock()

```
void Game::drawBackBlock ( )
```

Dessine un block.

Définition à la ligne 49 du fichier Game.cpp.

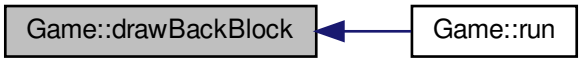
Références getBackBlock().

Référencé par run().

```
50 {
51     Rectangle back = getBackBlock();
52     DrawRectangleRounded(back, 0.5, 4, {255, 161, 0, 150});
53     DrawText("Back", back.x + back.width * 0.20, back.y + back.height * 0.2, 14, GRAY);
54 
55     if (mouseOnBack)
56         DrawRectangleRoundedLines(back, 0.5, 4, 5, RED);
57 
58     else
59         DrawRectangleRoundedLines(back, 0.5, 4, 5, ORANGE);
60 }
```

Voici le graphe d'appel pour cette fonction :

Voici le graphe des appelants de cette fonction :



4.4.3.3 getBackBlock()

```
Rectangle Game::getBackBlock ( )
```

Retourne une représentation Graphique.

Renvoie

Rectangle

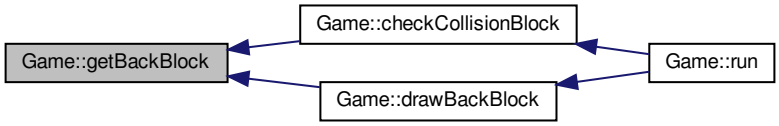
Définition à la ligne 20 du fichier Game.cpp.

Références State : :W_HEIGHT, et State : :W_WIDTH.

Référencé par checkCollisionBlock(), et drawBackBlock().

```
21 {
22     Rectangle back = {W_WIDTH * (float)0.01, W_HEIGHT * (float)0.01,
      W_WIDTH * (float)0.07, W_HEIGHT * (float)0.03};
23     return back;
24 }
```

Voici le graphe des appelants de cette fonction :



4.4.3.4 isBack()

```
void Game::isBack (
            App * app )
```

4.4.3.5 run()

```
void Game::run (
            App * app )  [virtual]
```

Lance l'exercice de simulation.

Paramètres

app	

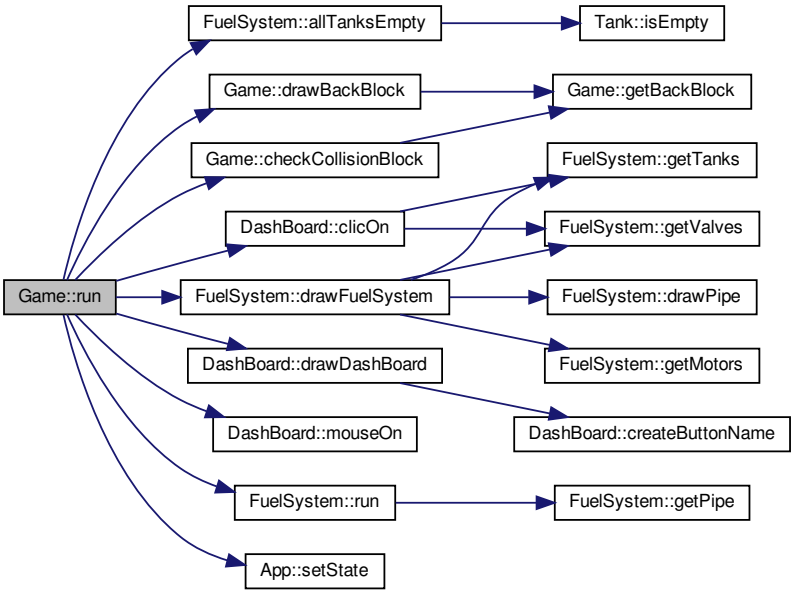
Implémente State.

Définition à la ligne 67 du fichier Game.cpp.

Références FuelSystem : :allTanksEmpty(), checkCollisionBlock(), DashBoard : :clicOn(), drawBackBlock(), Dash↩ Board : :drawDashBoard(), FuelSystem : :drawFuelSystem(), DashBoard : :mouseOn(), FuelSystem : :run(), App↩ : :setState(), State : :W_HEIGHT, et State : :W_WIDTH.

```
68 {
69     if (clicOnBack)
70         app->setState(new Menu);
71 
72     checkCollisionBlock();
73     drawBackBlock();
74 
75     if (!m_fs.allTanksEmpty())
76     {
77         m_db.clicOn(&m_fs);
78 
79         m_fs.run();
80 
81         m_fs.drawFuelSystem();
82 
83         m_db.mouseOn();
84         m_db.drawDashBoard();
85     }
86     else
87         DrawText("Game Over", W_WIDTH * 0.15, W_HEIGHT * 0.4, 100, RED);
88 }
```

Voici le graphe d'appel pour cette fonction :



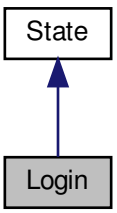
La documentation de cette classe a été générée à partir des fichiers suivants :

- include/Game.h
- src/Game.cpp

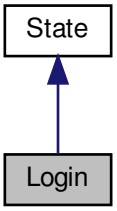
4.5 Référence de la classe Login

```
#include <Login.h>
```

Graphe d'héritage de Login :



Graphe de collaboration de Login :



Fonctions membres publiques

- Login ()
  Construct a new Login : : Login object.
- void checkCollisionBlock ()
  Saisie.
- Rectangle getIdBlock ()
  Champs de saisie pour l'id.
- Rectangle getPasswordBlock ()
  Champs de saisie pour le mot de passe.
- Rectangle getEnterBlock ()
  Validation du login.
- Rectangle getBackBlock ()
  Boutons Back.
- void drawIdBlock ()
  Information de l'user.
- void drawPasswordBlock ()
  Affiche le mot de passe.
- void drawEnterBlock ()
  Valide ou pas.
- void drawBackBlock ()
  Dessine le bouton back.
- virtual void run (App ∗app)
  Run la fenetre de login.

Membres hérités additionnels

4.5.1 Description détaillée

Définition à la ligne 9 du fichier Login.h.

4.5.2 Documentation des constructeurs et destructeur

4.5.2.1 Login()

```
Login::Login ( )
```

Construct a new Login : : Login object.

Définition à la ligne 12 du fichier Login.cpp.

```
12              : State(400, 600, (char *)"Login")
13 {
14 }
```

4.5.3 Documentation des fonctions membres

4.5.3.1 checkCollisionBlock()

```
void Login::checkCollisionBlock ( )
```

Saisie.

Définition à la ligne 64 du fichier Login.cpp.

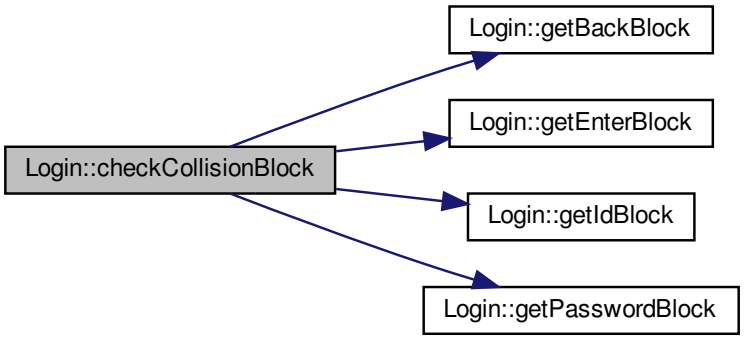
Références getBackBlock(), getEnterBlock(), getIdBlock(), et getPasswordBlock().

Référencé par run().

```
65 {
66     std::string m_Log;
67     std::string line;
68 
69     if (CheckCollisionPointRec(GetMousePosition(), getBackBlock()))
70     {
71         mouseOnBack = true;
72 
73         if (IsMouseButtonReleased(MOUSE_LEFT_BUTTON))
74         {
75             clicOnBack = true;
76         }
77     }
78     else
79         mouseOnBack = false;
80 
81     if (CheckCollisionPointRec(GetMousePosition(), getIdBlock()))
82     {
83         mouseOnId = true;
84 
85         int key = GetKeyPressed();
86 
87         if ((key >= 32) && (key <= 125) && (m_id.size() < MAX_INPUT_CHARS))
88         {
89             m_id.push_back((char)key);
90         }
91 
92         if (IsKeyPressed(KEY_BACKSPACE))
93         {
94             if (!m_id.empty())
95             {
96                 m_id.pop_back();
97             }
98         }
99     }
100 
101      else
102          mouseOnId = false;
103 
```

```
104     if (CheckCollisionPointRec(GetMousePosition(), getPasswordBlock()))
105     {
106         mouseOnPassword = true;
107 
108         int key = GetKeyPressed();
109 
110         if ((key >= 32) && (key <= 125) && (m_pass.size() < MAX_INPUT_CHARS))
111         {
112             m_pass.push_back((char)key);
113         }
114 
115         if (IsKeyPressed(KEY_BACKSPACE))
116         {
117             if (!m_pass.empty())
118             {
119                 m_pass.pop_back();
120             }
121         }
122     }
123     else
124         mouseOnPassword = false;
125 
126     if (CheckCollisionPointRec(GetMousePosition(), getEnterBlock()) && IsMouseButtonReleased(
    MOUSE_LEFT_BUTTON))
127     {
128         mouseOnEnter = true;
129         std::ifstream bdd("userBDD.txt");
130 
131         if (bdd)
132         {
133             m_Log = m_id + " " + m_pass;
134 
135             while (getline(bdd, line))
136             {
137                 if (m_Log.compare(line) == 0)
138                 {
139                     clicOnEnter = true;
140                 }
141 
142                 else
143                 {
144                     connexion = false;
145                 }
146             }
147         }
148     }
149     else
150         mouseOnEnter = false;
151 
152     if (mouseOnId)
153         framesCounterLogin++;
154     else
155         framesCounterLogin = 0;
156 
157     if (mouseOnPassword)
158         framesCounterPassword++;
159     else
160         framesCounterPassword = 0;
161 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :

4.5.3.2 drawBackBlock()

```
void Login::drawBackBlock ( )
```

Dessine le bouton back.

Définition à la ligne 246 du fichier Login.cpp.

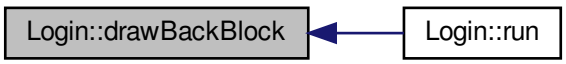
Références getBackBlock().

Référencé par run().

```
247 {
248     Rectangle back = getBackBlock();
249     DrawRectangleRounded(back, 0.5, 4, {255, 161, 0, 150});
250     DrawText("Exit", back.x + back.width * 0.25, back.y + back.height * 0.27, 20, GRAY);
251 
252     if (mouseOnBack)
253         DrawRectangleRoundedLines(back, 0.5, 4, 5, RED);
254 
255     else
256         DrawRectangleRoundedLines(back, 0.5, 4, 5, ORANGE);
257 }
```

Voici le graphe d'appel pour cette fonction :

Voici le graphe des appelants de cette fonction :



4.5.3.3 drawEnterBlock()

```
void Login::drawEnterBlock ( )
```

Valide ou pas.

Définition à la ligne 225 du fichier Login.cpp.

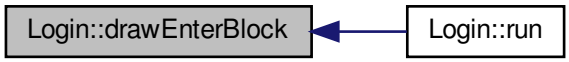
Références getEnterBlock(), et State : :W_WIDTH.

Référencé par run().

```
226 {
227     Rectangle enter = getEnterBlock();
228     DrawRectangleRec(enter, LIGHTGRAY);
229     DrawText("Enter", enter.x + 30, enter.y + 20, 15, GRAY);
230 
231     if (CheckCollisionPointRec(GetMousePosition(), enter))
232         DrawRectangleLines(enter.x, enter.y, enter.width, enter.height, BLUE);
233     else
234         DrawRectangleLines(enter.x, enter.y, enter.width, enter.height, DARKGRAY);
235 
236     if (!connexion)
237     {
238         DrawText("Error: Wrongs Logs", W_WIDTH / 2 - (225 / 2), 480, 20, RED);
239     }
240 }
```

Voici le graphe d'appel pour cette fonction :

Voici le graphe des appelants de cette fonction :



4.5.3.4 drawIdBlock()

```
void Login::drawIdBlock ( )
```

Information de l'user.

Définition à la ligne 167 du fichier Login.cpp.

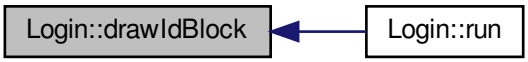
Références getIdBlock(), et State : :W_WIDTH.

Référencé par run().

```
168 {
169     Rectangle id = getIdBlock();
170     DrawText("ID", W_WIDTH / 2 - (225 / 2), 160, 20, GRAY);
171     DrawRectangleRec(id, LIGHTGRAY);
172 
173     if (mouseOnId)
174         DrawRectangleLines(id.x, id.y, id.width, id.height, BLUE);
175     else
176         DrawRectangleLines(id.x, id.y, id.width, id.height, DARKGRAY);
177 
178     DrawText(m_id.c_str(), id.x + 5, id.y + 8, 40, BLUE);
179 
180     if (mouseOnId)
181     {
182         if (m_id.size() < MAX_INPUT_CHARS)
183         {
184             if (((framesCounterLogin / 20) % 2) == 0)
185                 DrawText("_", id.x + 8 + MeasureText(m_id.c_str(), 40), id.y + 12, 40, BLUE);
186         }
187         else
188             DrawText("Your name is too long", W_WIDTH / 2 - (225 / 2), 260, 15, GRAY);
189     }
190 }
```

Voici le graphe d'appel pour cette fonction :

Voici le graphe des appelants de cette fonction :



4.5.3.5 drawPasswordBlock()

```
void Login::drawPasswordBlock ( )
```

Affiche le mot de passe.

Définition à la ligne 196 du fichier Login.cpp.

Références getPasswordBlock(), et State : :W_WIDTH.

Référencé par run().

```
197 {
198     Rectangle password = getPasswordBlock();
199     DrawText("PASSWORD", W_WIDTH / 2 - (225 / 2), 280, 20, GRAY);
200     DrawRectangleRec(password, LIGHTGRAY);
201 
202     if (mouseOnPassword)
203         DrawRectangleLines(password.x, password.y, password.width, password.height, BLUE);
204     else
205         DrawRectangleLines(password.x, password.y, password.width, password.height, DARKGRAY);
206 
207     DrawText(m_pass.c_str(), password.x + 5, password.y + 8, 40, BLUE);
208 
209     if (mouseOnPassword)
210     {
211         if (m_pass.size() < MAX_INPUT_CHARS)
212         {
213             if (((framesCounterPassword / 20) % 2) == 0)
214                 DrawText("_", password.x + 8 + MeasureText(m_pass.c_str(), 40), password.y + 12, 40, BLUE);
215         }
216         else
217             DrawText("Your password is too long", W_WIDTH / 2 - (225 / 2), 360, 15, GRAY);
218     }
219 }
```

Voici le graphe d'appel pour cette fonction :

Voici le graphe des appelants de cette fonction :

4.5.3.6 getBackBlock()

```
Rectangle Login::getBackBlock ( )
```

Boutons Back.

Renvoie

Rectangle

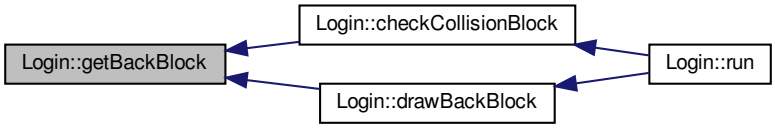
Définition à la ligne 54 du fichier Login.cpp.

Références State : :W_HEIGHT, et State : :W_WIDTH.

Référencé par checkCollisionBlock(), et drawBackBlock().

```
55 {
56     Rectangle exit = {W_WIDTH * (float)0.02, W_HEIGHT * (float)0.02,
      W_WIDTH * (float)0.2, W_HEIGHT * (float)0.05};
57     return exit;
58 }
```

Voici le graphe des appelants de cette fonction :



4.5.3.7 getEnterBlock()

```
Rectangle Login::getEnterBlock ( )
```

Validation du login.

Renvoie

Rectangle

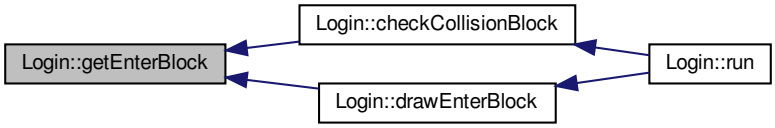
Définition à la ligne 43 du fichier Login.cpp.

Références State : :W_WIDTH.

Référencé par checkCollisionBlock(), et drawEnterBlock().

```
44 {
45     Rectangle enter = {W_WIDTH / 2 - (100 / 2), 400, 100, 50};
46     return enter;
47 }
```

Voici le graphe des appelants de cette fonction :



4.5.3.8 getIdBlock()

```
Rectangle Login::getIdBlock ( )
```

Champs de saisie pour l'id.

Renvoie

Rectangle

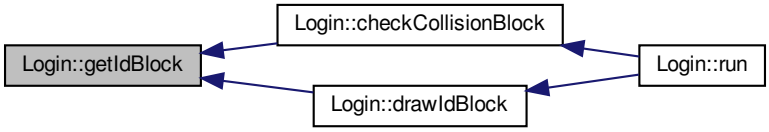
Définition à la ligne 21 du fichier Login.cpp.

Références State : :W_WIDTH.

Référencé par checkCollisionBlock(), et drawIdBlock().

```
22 {
23     Rectangle id = {W_WIDTH / 2 - (225 / 2), 180, 225, 50};
24     return id;
25 }
```

Voici le graphe des appelants de cette fonction :



4.5.3.9 getPasswordBlock()

```
Rectangle Login::getPasswordBlock ( )
```

Champs de saisie pour le mot de passe.

Renvoie

Rectangle

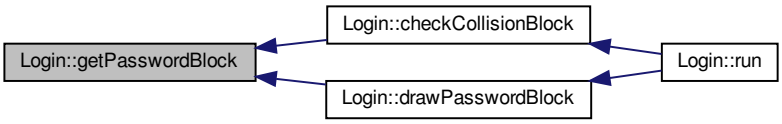
Définition à la ligne 32 du fichier Login.cpp.

Références State : :W_WIDTH.

Référencé par checkCollisionBlock(), et drawPasswordBlock().

```
33 {
34     Rectangle password = {W_WIDTH / 2 - (225 / 2), 300, 225, 50};
35     return password;
36 }
```

Voici le graphe des appelants de cette fonction :



4.5.3.10 run()

```
void Login::run (
            App * app )  [virtual]
```

Run la fenetre de login.

Paramètres

app	

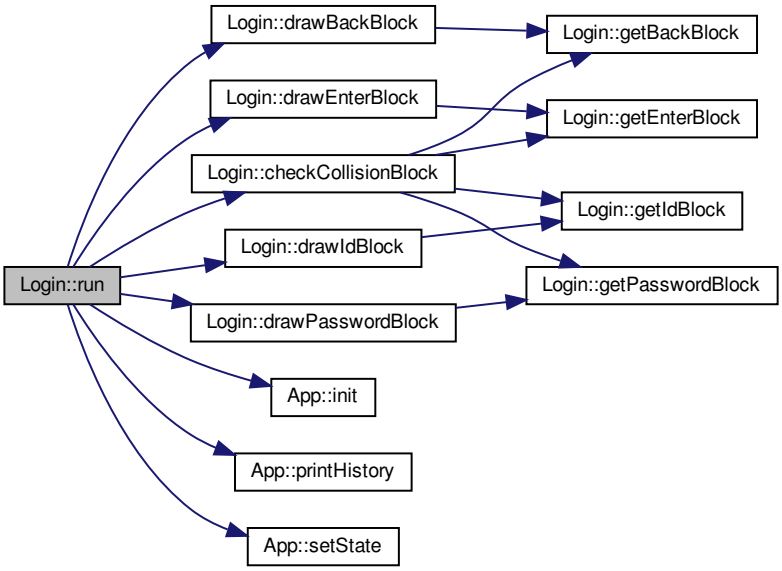
Implémente State.

Définition à la ligne 264 du fichier Login.cpp.

Références checkCollisionBlock(), drawBackBlock(), drawEnterBlock(), drawIdBlock(), drawPasswordBlock(), App : :init(), App : :printHistory(), et App : :setState().

```
265 {
266     if (clicOnBack)
267     {
268         exit(0);
269     }
270     else if (clicOnEnter)
271     {
272         app->init(this->m_id);
273         app->printHistory();
274         app->setState(new Menu);
275     }
276     checkCollisionBlock();
277 
278     drawIdBlock();
279     drawPasswordBlock();
280     drawEnterBlock();
281     drawBackBlock();
282 }
```

Voici le graphe d'appel pour cette fonction :



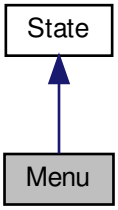
La documentation de cette classe a été générée à partir des fichiers suivants :

- include/Login.h
- src/Login.cpp

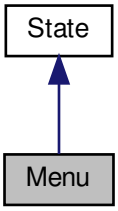
4.6 Référence de la classe Menu

`#include <Menu.h>`

Graphe d'héritage de Menu :



Graphe de collaboration de Menu :



Fonctions membres publiques

- Menu ()
  Construct a new Menu : : Menu object.
- Rectangle getStartBlock ()
  Représentation du bouton de démarrage.
- Rectangle getBackBlock ()
  Représentation du bouton back.
- void checkCollisionBlock ()
  Vérifie les cliques de la souris.
- void drawStartBlock ()
  Dessine un bouton Start.
- void drawBackBlock ()
  Dessine un bouton back.
- virtual void run (App ∗app)
  Opération possible dans le menu.

Membres hérités additionnels

4.6.1 Description détaillée

Définition à la ligne 6 du fichier Menu.h.

4.6.2 Documentation des constructeurs et destructeur

4.6.2.1 Menu()

```
Menu::Menu ( )
```

Construct a new Menu : : Menu object.

Définition à la ligne 9 du fichier Menu.cpp.

Références State : :W_HEIGHT, State : :W_TITLE, et State : :W_WIDTH.

```
9             : State(400, 600, (char *)"Menu")
10 {
11     SetWindowSize((int)W_WIDTH, (int)W_HEIGHT);
12     SetWindowTitle(W_TITLE);
13 }
```

4.6.3 Documentation des fonctions membres

4.6.3.1 checkCollisionBlock()

```
void Menu::checkCollisionBlock ( )
```

Vérifie les cliques de la souris.

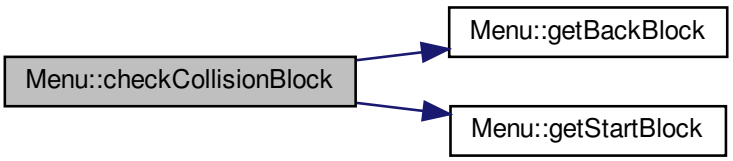
Définition à la ligne 41 du fichier Menu.cpp.

Références getBackBlock(), et getStartBlock().

Référencé par run().

```
42 {
43     if (CheckCollisionPointRec(GetMousePosition(), getBackBlock()))
44     {
45         mouseOnBack = true;
46 
47         if (IsMouseButtonReleased(MOUSE_LEFT_BUTTON))
48         {
49             clickOnBack = true;
50         }
51     }
52     else
53         mouseOnBack = false;
54 
55     if (CheckCollisionPointRec(GetMousePosition(), getStartBlock()))
56     {
57         mouseOnStart = true;
58 
59         if (IsMouseButtonReleased(MOUSE_LEFT_BUTTON))
60         {
61             clicOnStart = true;
62         }
63     }
64     else
65         mouseOnStart = false;
66 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :

4.6.3.2 drawBackBlock()

```
void Menu::drawBackBlock ( )
```

Dessine un bouton back.

Définition à la ligne 89 du fichier Menu.cpp.

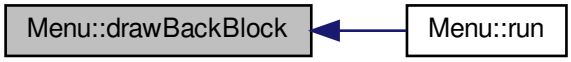
Références getBackBlock().

Référencé par run().

```
90 {
91     Rectangle back = getBackBlock();
92     DrawRectangleRounded(back, 0.5, 4, {255, 161, 0, 150});
93     DrawText("Log out", back.x + back.width * 0.05, back.y + back.height * 0.27, 20, GRAY);
94 
95     if (mouseOnBack)
96         DrawRectangleRoundedLines(back, 0.5, 4, 5, RED);
97 
98     else
99         DrawRectangleRoundedLines(back, 0.5, 4, 5, ORANGE);
100 }
```

Voici le graphe d'appel pour cette fonction :

Voici le graphe des appelants de cette fonction :



4.6.3.3 drawStartBlock()

```
void Menu::drawStartBlock ( )
```

Dessine un bouton Start.

Définition à la ligne 72 du fichier Menu.cpp.

Références getStartBlock().

Référencé par run().

```
73 {
74     Rectangle start = getStartBlock();
75     DrawRectangleRec(start, LIGHTGRAY);
76     DrawText("Start", start.x + start.width * 0.22, start.y + start.height * 0.25, 40, GRAY);
77 
78     if (mouseOnStart)
79         DrawRectangleLinesEx(start, 2, BLUE);
80 
81     else
82         DrawRectangleLinesEx(start, 2, DARKGRAY);
83 }
```

Voici le graphe d'appel pour cette fonction :

Voici le graphe des appelants de cette fonction :

4.6.3.4 getBackBlock()

```
Rectangle Menu::getBackBlock ( )
```

Représentation du bouton back.

Renvoie

Rectangle

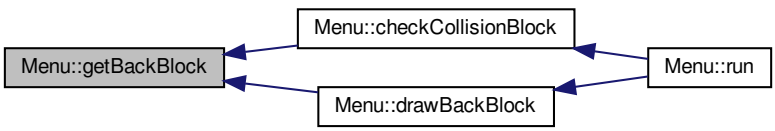
Définition à la ligne 31 du fichier Menu.cpp.

Références State : :W_HEIGHT, et State : :W_WIDTH.

Référencé par checkCollisionBlock(), et drawBackBlock().

```
32 {
33     Rectangle back = {W_WIDTH * (float)0.02, W_HEIGHT * (float)0.02,
      W_WIDTH * (float)0.2, W_HEIGHT * (float)0.05};
34     return back;
35 }
```

Voici le graphe des appelants de cette fonction :



4.6.3.5 getStartBlock()

```
Rectangle Menu::getStartBlock ( )
```

Représentation du bouton de démarrage.

Renvoie

Rectangle

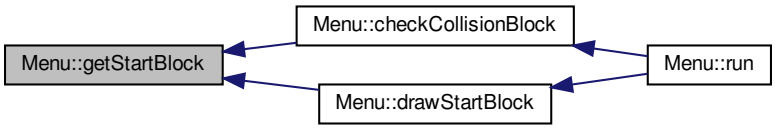
Définition à la ligne 20 du fichier Menu.cpp.

Références State : :W_HEIGHT, et State : :W_WIDTH.

Référencé par checkCollisionBlock(), et drawStartBlock().

```
21 {
22     Rectangle start = {((W_WIDTH / 2) - (W_WIDTH / 2) / 2),
      W_HEIGHT * (float)0.25, (W_WIDTH / 2), W_HEIGHT / 8};
23     return start;
24 }
```

Voici le graphe des appelants de cette fonction :



4.6.3.6 run()

```
void Menu::run (
            App * app )  [virtual]
```

Opération possible dans le menu.

Paramètres

app	

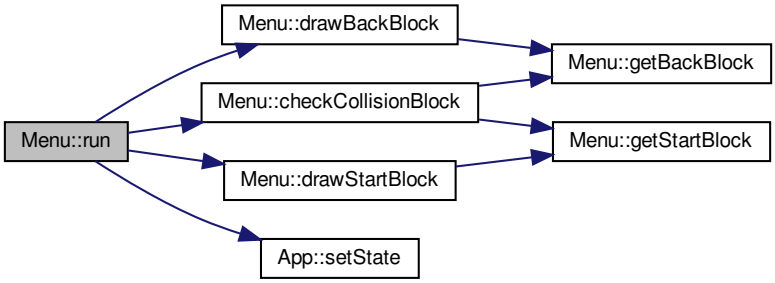
Implémente State.

Définition à la ligne 107 du fichier Menu.cpp.

Références checkCollisionBlock(), drawBackBlock(), drawStartBlock(), et App : :setState().

```
108 {
109     if (clicOnStart)
110     {
111         app->setState(new Game);
112     }
113     else if (clickOnBack)
114     {
115         app->setState(new Login);
116     }
117     checkCollisionBlock();
118     drawStartBlock();
119     drawBackBlock();
120 }
```

Voici le graphe d'appel pour cette fonction :



La documentation de cette classe a été générée à partir des fichiers suivants :
- include/Menu.h
- src/Menu.cpp

4.7 Référence de la classe Motor

`#include <Motor.h>`

Fonctions membres publiques

- void initMemberGl ()
  Initialisation des Moteurs (graphiquement)
- Rectangle getMotorBlock ()
  Représentation d'un Moteur.
- void drawState ()
  Représentation des états d'un moteur.
- void drawMotor ()
  Dessine un moteur.
- Motor ()
  Construct a new Motor : : Motor object.
- Motor (std : :string, Pump ∗)
  Construct a new Motor : : Motor object.
- StateMotor getStateMotor ()
- void setPump (Pump ∗)
  Définie une pompe au moteur.
- Pump getPump ()
  Retourne une pompe lié au moteur.
- void beingFed ()

Moteur alimenté
— void beingNotFed ()
Moteur en panne.
— bool isFed ()
Connaitre l'état du moteur.
— void update ()
Mise à jour de l'état du moteur.

4.7.1 Description détaillée

Définition à la ligne 9 du fichier Motor.h.

4.7.2 Documentation des constructeurs et destructeur

4.7.2.1 Motor() [1/2]

```
Motor::Motor ( )
```

Construct a new Motor : : Motor object.

Définition à la ligne 8 du fichier Motor.cpp.

```
9 {
10 }
```

4.7.2.2 Motor() [2/2]

```
Motor::Motor (
            std::string name,
            Pump * origin )
```

Construct a new Motor : : Motor object.

Paramètres

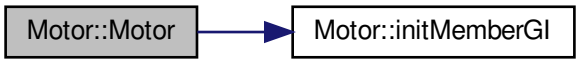
name	
origin	

Définition à la ligne 18 du fichier Motor.cpp.

Références initMemberGI().

```
18                                      : m_Name(name), m_origin(origin), m_stateMotor(FED)
19 {
20     initMemberGI();
21 };
```

Voici le graphe d'appel pour cette fonction :



4.7.3 Documentation des fonctions membres

4.7.3.1 beingFed()

```
void Motor::beingFed ( )
```

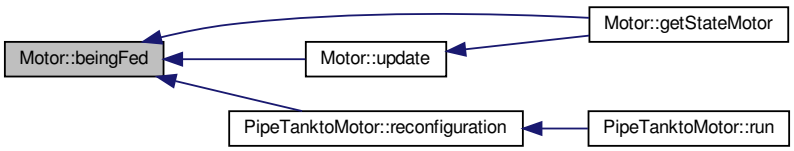
Moteur alimenté

Définition à la ligne 110 du fichier Motor.cpp.

Référencé par getStateMotor(), PipeTanktoMotor : :reconfiguration(), et update().

```
111 {
112     m_stateMotor = FED;
113 }
```

Voici le graphe des appelants de cette fonction :



4.7.3.2 beingNotFed()

```
void Motor::beingNotFed ( )
```

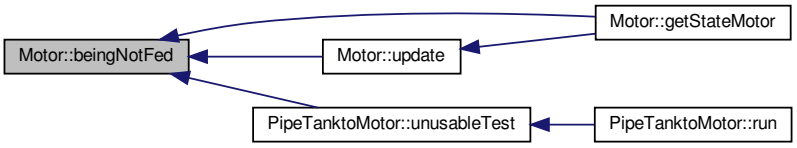
Moteur en panne.

Définition à la ligne 119 du fichier Motor.cpp.

Référencé par getStateMotor(), PipeTanktoMotor : :unusableTest(), et update().

```
120 {
121     m_stateMotor = NFED;
122 }
```

Voici le graphe des appelants de cette fonction :



4.7.3.3 drawMotor()

```
void Motor::drawMotor ( )
```

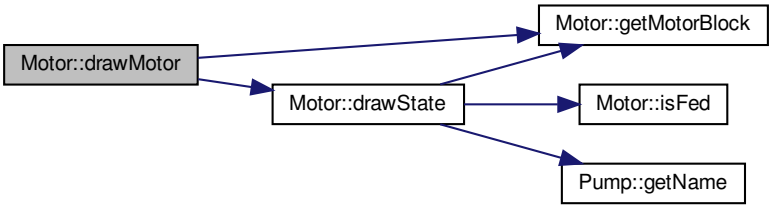
Dessine un moteur.

Définition à la ligne 79 du fichier Motor.cpp.

Références drawState(), et getMotorBlock().

```
80 {
81     DrawRectangleRec(getMotorBlock(), M_COLOR);
82     DrawRectangleLinesEx(getMotorBlock(), 4, DARKGRAY);
83     drawState();
84 }
```

Voici le graphe d'appel pour cette fonction :



4.7.3.4 drawState()

```
void Motor::drawState ( )
```

Représentation des états d'un moteur.

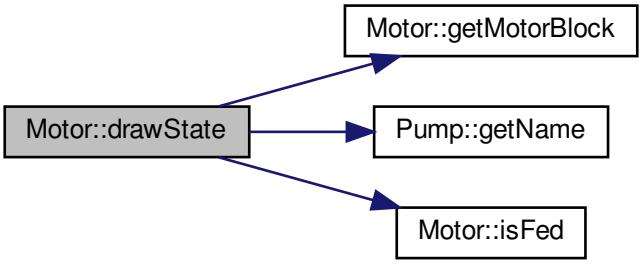
Définition à la ligne 58 du fichier Motor.cpp.

Références getMotorBlock(), Pump : :getName(), et isFed().

Référencé par drawMotor().

```
59 {
60     Rectangle motor = getMotorBlock();
61 
62     if (isFed())
63     {
64         DrawText("Fed by", motor.x + motor.width * 0.07, motor.y + motor.height * 0.25, 20, GREEN);
65         DrawText(m_origin->getName().c_str(), motor.x + motor.width * 0.3, motor.y + motor.height *
       0.60, 20, GREEN);
66     }
67 
68     else
69     {
70         DrawText("Not", motor.x + motor.width * 0.13, motor.y + motor.height * 0.25, 20, RED);
71         DrawText("Fed", motor.x + motor.width * 0.43, motor.y + motor.height * 0.60, 20, RED);
72     }
73 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :

4.7.3.5 getMotorBlock()

```
Rectangle Motor::getMotorBlock ( )
```

Représentation d'un Moteur.

Renvoie

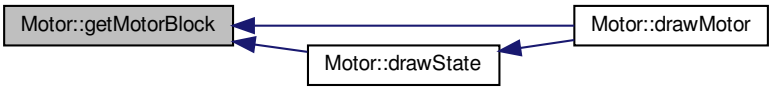
Rectangle

Définition à la ligne 48 du fichier Motor.cpp.

Référencé par drawMotor(), et drawState().

```
49 {
50     Rectangle t_rec = {(float)M_POSX, (float)M_POSY, (float)M_WIDTH, (float)M_HEIGHT};
51     return t_rec;
52 }
```

Voici le graphe des appelants de cette fonction :



4.7.3.6 getPump()

```
Pump Motor::getPump ( )
```

Retourne une pompe lié au moteur.

Renvoie

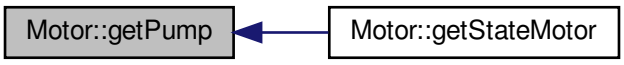
Pump

Définition à la ligne 101 du fichier Motor.cpp.

Référencé par getStateMotor().

```
102 {
103     return *m_origin;
104 }
```

Voici le graphe des appelants de cette fonction :



4.7.3.7 getStateMotor()

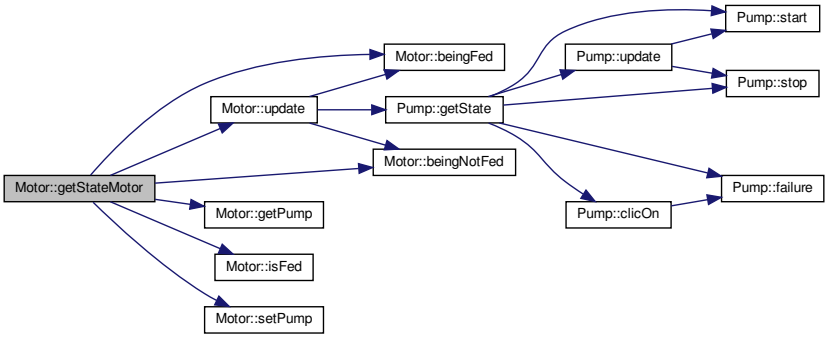
```
StateMotor Motor::getStateMotor ( )  [inline]
```

Définition à la ligne 35 du fichier Motor.h.

Références beingFed(), beingNotFed(), getPump(), isFed(), setPump(), et update().

```
35 { return m_stateMotor; }
```

Voici le graphe d'appel pour cette fonction :



4.7.3.8 initMemberGI()

```
void Motor::initMemberGI ( )
```

Initialisation des Moteurs (graphiquement)

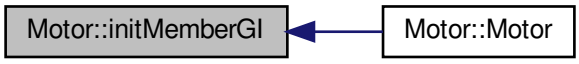
Définition à la ligne 27 du fichier Motor.cpp.

Références FL_WIDTH.

Référencé par Motor().

```
28 {
29     if (m_Name.compare("M1") == 0)
30     {
31         M_POSX = ((FL_WIDTH / 8) + (FL_WIDTH / 6) / 2 - M_WIDTH / 2);
32     }
33     else if (m_Name.compare("M2") == 0)
34     {
35         M_POSX = (((FL_WIDTH / 2) - ((FL_WIDTH / 8) / 2)) + (
      FL_WIDTH / 8) / 2 - M_WIDTH / 2);
36     }
37     else if (m_Name.compare("M3") == 0)
38     {
39         M_POSX = ((FL_WIDTH - ((FL_WIDTH / 8) + (FL_WIDTH / 6))) + (
      FL_WIDTH / 6) / 2 - M_WIDTH / 2);
40     }
41 }
```

Voici le graphe des appelants de cette fonction :



4.7.3.9 isFed()

```
bool Motor::isFed ( )
```

Connaitre l'état du moteur.

Renvoie

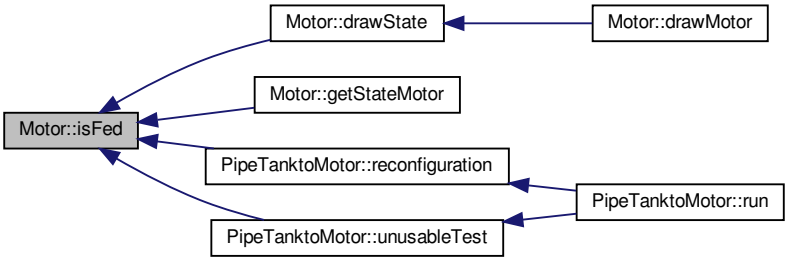
true
false

Définition à la ligne 130 du fichier Motor.cpp.

Référencé par drawState(), getStateMotor(), PipeTanktoMotor : :reconfiguration(), et PipeTanktoMotor : :unusable↩ Test().

```
131 {
132     if (m_stateMotor == FED)
133         return true;
134     return false;
135 }
```

Voici le graphe des appelants de cette fonction :



4.7.3.10 setPump()

```
void Motor::setPump (
            Pump * pump )
```

Définie une pompe au moteur.

Paramètres

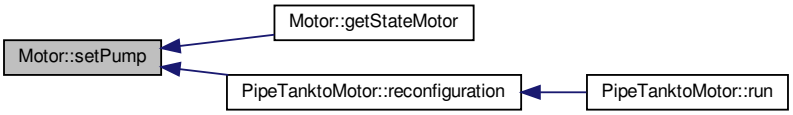
pump	

Définition à la ligne 91 du fichier Motor.cpp.

Référencé par getStateMotor(), et PipeTanktoMotor : :reconfiguration().

```
92 {
93     m_origin = pump;
94 }
```

Voici le graphe des appelants de cette fonction :



4.7.3.11 update()

```
void Motor::update ( )
```

Mise à jour de l'état du moteur.

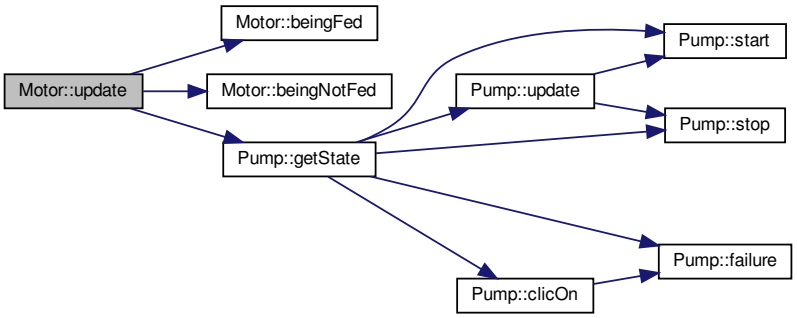
Définition à la ligne 141 du fichier Motor.cpp.

Références beingFed(), beingNotFed(), et Pump : :getState().

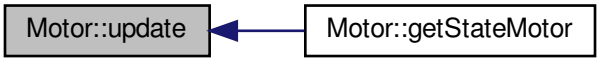
Référencé par getStateMotor().

```
142 {
143     if (m_origin->getState() != 0)
144     {
145         beingNotFed();
146     }
147 
148     else
149     {
150         beingFed();
151     }
152 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :

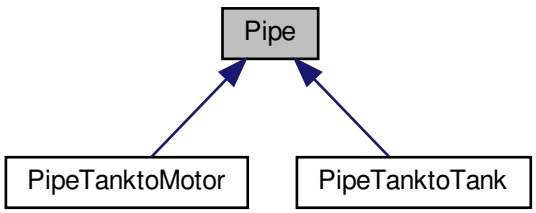


La documentation de cette classe a été générée à partir des fichiers suivants :
- include/Motor.h
- src/Motor.cpp

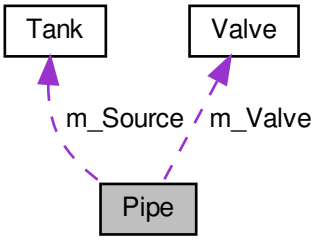
4.8 Référence de la classe Pipe

`#include <Pipe.h>`

Graphe d'héritage de Pipe :



Graphe de collaboration de Pipe :



Fonctions membres publiques

- Pipe (Tank *, Valve *)
  Construct a new Pipe : : Pipe object.
- virtual ~Pipe ()
  Destroy the Pipe : : Pipe object.
- virtual void reconfiguration ()=0
- virtual void run ()
  Le tube délivre juste le fuel.

Attributs protégés

- Tank * m_Source
- Valve * m_Valve

4.8.1 Description détaillée

Définition à la ligne 6 du fichier Pipe.h.

4.8.2 Documentation des constructeurs et destructeur

4.8.2.1 Pipe()

```
Pipe::Pipe (
            Tank * source,
            Valve * valve )
```

Construct a new Pipe : : Pipe object.

Paramètres

source	
valve	

Définition à la ligne 9 du fichier Pipe.cpp.

```
9                                       : m_Source(source), m_Valve(valve)
10 {
11 }
```

4.8.2.2 ~Pipe()

```
Pipe::~Pipe ( )  [virtual]
```

Destroy the Pipe : : Pipe object.

Définition à la ligne 25 du fichier Pipe.cpp.

```
26 {
27 }
```

4.8.3 Documentation des fonctions membres

4.8.3.1 reconfiguration()

```
virtual void Pipe::reconfiguration ( )  [pure virtual]
```

Implémenté dans PipeTanktoMotor, et PipeTanktoTank.

4.8.3.2 run()

```
void Pipe::run ( )  [virtual]
```

Le tube délivre juste le fuel.

Réimplémentée dans PipeTanktoMotor, et PipeTanktoTank.

Définition à la ligne 17 du fichier Pipe.cpp.

```
18 {
19 }
```

4.8.4 Documentation des données membres

4.8.4.1 m_Source

```
Tank* Pipe::m_Source  [protected]
```

Définition à la ligne 9 du fichier Pipe.h.

Référencé par PipeTanktoTank : :reconfiguration(), PipeTanktoMotor : :reconfiguration(), et PipeTanktoMotor↩ : :unusableTest().

4.8.4.2 m_Valve

```
Valve* Pipe::m_Valve  [protected]
```

Définition à la ligne 10 du fichier Pipe.h.

Référencé par PipeTanktoMotor : :reconfiguration(), PipeTanktoTank : :run(), et PipeTanktoMotor : :unusableTest().

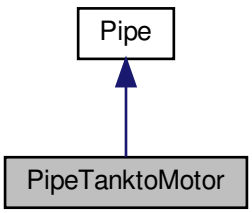
La documentation de cette classe a été générée à partir des fichiers suivants :

- include/Pipe.h
- src/Pipe.cpp

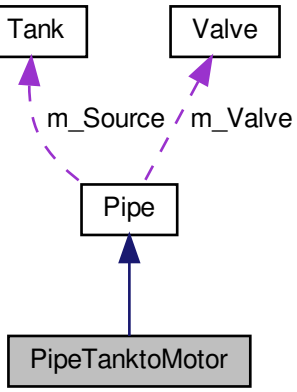
4.9 Référence de la classe PipeTanktoMotor

```
#include <PipeTankToMotor.h>
```

Graphe d'héritage de PipeTanktoMotor :



Graphe de collaboration de PipeTanktoMotor :



Fonctions membres publiques

- PipeTanktoMotor (Tank ∗, Valve ∗, Valve ∗, Motor ∗)
  Construct a new Pipe Tankto Motor : : Pipe Tankto Motor object.
- void unusableTest ()
  Cas de base.
- void updateMotor ()
- void reconfiguration ()
  Cas custome.
- virtual void run ()
  Pompe to moteur.

Membres hérités additionnels

4.9.1 Description détaillée

Définition à la ligne 7 du fichier PipeTankToMotor.h.

4.9.2 Documentation des constructeurs et destructeur

4.9.2.1 PipeTanktoMotor()

```
PipeTanktoMotor::PipeTanktoMotor (
            Tank * source,
            Valve * valve1,
            Valve * valve2,
            Motor * motor )
```

Construct a new Pipe Tankto Motor : : Pipe Tankto Motor object.

Paramètres

source	
valve1	
valve2	
motor	

Définition à la ligne 12 du fichier PipeTankToMotor.cpp.

```
12                                                                                  :
      Pipe(source, valve1), m_Valve2(valve2), m_Motor(motor)
13 {
14 }
```

4.9.3 Documentation des fonctions membres

4.9.3.1 reconfiguration()

```
void PipeTanktoMotor::reconfiguration ( )  [virtual]
```

Cas custome.

Implémente Pipe.

Définition à la ligne 53 du fichier PipeTankToMotor.cpp.

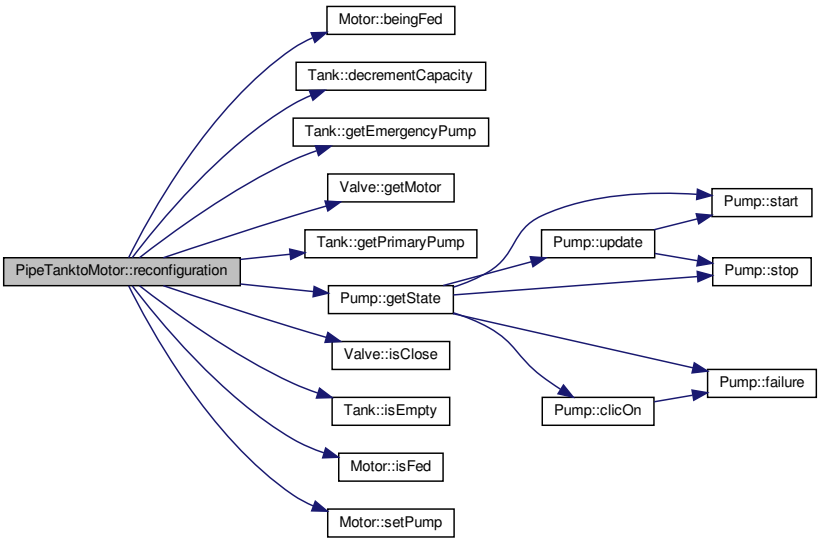
Références Motor : :beingFed(), Tank : :decrementCapacity(), Tank : :getEmergencyPump(), Valve : :getMotor(), Tank : :getPrimaryPump(), Pump : :getState(), Valve : :isClose(), Tank : :isEmpty(), Motor : :isFed(), Pipe : :m_↩ Source, Pipe : :m_Valve, et Motor : :setPump().

Référencé par run().

```
54 {
55     //Pompe secondaire prend le relai
56     if (!m_Source->isEmpty() && m_Source->getPrimaryPump().
      getState() == 2 && m_Source->getEmergencyPump().
      getState() == 0 && !m_Valve->isClose() && !m_Valve2->
      isClose())
57     {
58         m_Source->decrementCapacity(1);
59 
60         if (!m_Motor->isFed())
61         {
62             m_Motor->setPump(&m_Source->getEmergencyPump());
63             m_Motor->beingFed();
64         }
65     }
66     //Pompe principale alimente d'autre moteur
67     else if (m_Source->getPrimaryPump().getState() == 0 &&
      m_Valve->isClose() && !m_Valve2->isClose())
68     {
69         m_Source->decrementCapacity(2);
70 
71         if (!m_Valve->getMotor(m_Source)->isFed())
72         {
73             m_Valve->getMotor(m_Source)->setPump(&
      m_Source->getPrimaryPump());
74             m_Valve->getMotor(m_Source)->beingFed();
75         }
76     }
77 
```

```
78      //Pompe principale alimente d'autre moteur
79      else if (m_Source->getPrimaryPump().getState() == 0 && !
        m_Valve->isClose() && m_Valve2->isClose())
80      {
81          m_Source->decrementCapacity(2);
82
83          if (!m_Valve2->getMotor(m_Source)->isFed())
84          {
85              m_Valve2->getMotor(m_Source)->setPump(&
        m_Source->getPrimaryPump());
86              m_Valve2->getMotor(m_Source)->beingFed();
87          }
88      }
89
90      //Pompe principale alimente tout les moteurs
91      else if (m_Source->getPrimaryPump().getState() == 0 &&
        m_Valve->isClose() && m_Valve2->isClose())
92      {
93          m_Source->decrementCapacity(3);
94
95          if (!m_Valve->getMotor(m_Source)->isFed())
96          {
97              m_Valve->getMotor(m_Source)->setPump(&
        m_Source->getPrimaryPump());
98              m_Valve->getMotor(m_Source)->beingFed();
99          }
100
101         else if (!m_Valve2->getMotor(m_Source)->isFed())
102         {
103             m_Valve2->getMotor(m_Source)->setPump(&
        m_Source->getPrimaryPump());
104             m_Valve2->getMotor(m_Source)->beingFed();
105         }
106     }
107
108     //Pompe secondaire alimente d'autre moteur
109     else if (m_Source->getEmergencyPump().getState() == 0 &&
        m_Valve->isClose() && !m_Valve2->isClose())
110     {
111         m_Source->decrementCapacity(2);
112
113         if (!m_Valve->getMotor(m_Source)->isFed())
114         {
115             m_Valve->getMotor(m_Source)->setPump(&
        m_Source->getEmergencyPump());
116             m_Valve->getMotor(m_Source)->beingFed();
117         }
118     }
119
120     //Pompe secondaire aliment d'autre moteur
121     else if (m_Source->getEmergencyPump().getState() == 0 && m_Valve2->
        isClose() && !m_Valve->isClose())
122     {
123         m_Source->decrementCapacity(2);
124
125         if (!m_Valve2->getMotor(m_Source)->isFed())
126         {
127             m_Valve2->getMotor(m_Source)->setPump(&
        m_Source->getEmergencyPump());
128             m_Valve2->getMotor(m_Source)->beingFed();
129         }
130     }
131
132     //Pompe secondaire aliment tout les moteurs
133     else if (m_Source->getEmergencyPump().getState() == 0 &&
        m_Valve->isClose() && m_Valve2->isClose())
134     {
135         m_Source->decrementCapacity(3);
136
137         if (!m_Valve->getMotor(m_Source)->isFed())
138         {
139             m_Valve->getMotor(m_Source)->setPump(&
        m_Source->getEmergencyPump());
140             m_Valve->getMotor(m_Source)->beingFed();
141         }
142
143         else if (!m_Valve2->getMotor(m_Source)->isFed())
144         {
145             m_Valve2->getMotor(m_Source)->setPump(&
        m_Source->getEmergencyPump());
146             m_Valve2->getMotor(m_Source)->beingFed();
147         }
148     }
149 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :

4.9.3.2 run()

```
void PipeTanktoMotor::run ( )  [virtual]
```

Pompe to moteur.

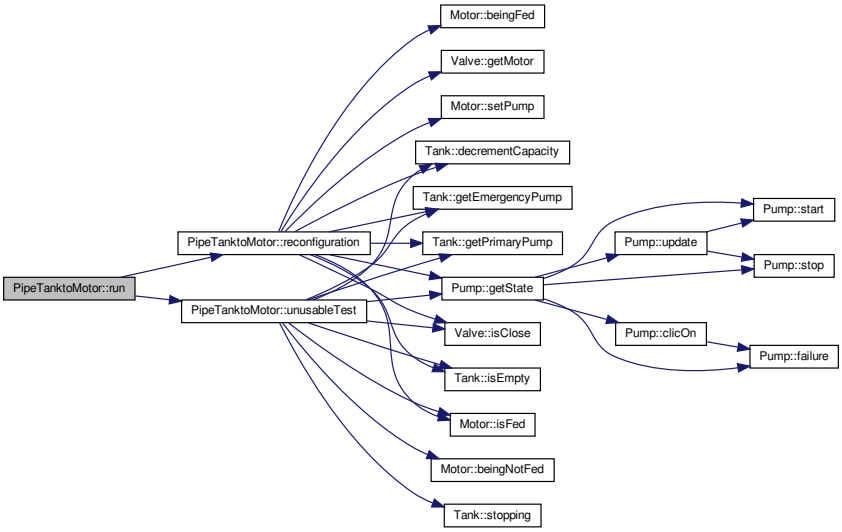
Réimplémentée à partir de Pipe.

Définition à la ligne 155 du fichier PipeTankToMotor.cpp.

Références reconfiguration(), et unusableTest().

```
156 {
157     unusableTest();
158     reconfiguration();
159 }
```

Voici le graphe d'appel pour cette fonction :



4.9.3.3 unusableTest()

```
void PipeTanktoMotor::unusableTest ( )
```

Cas de base.

Définition à la ligne 20 du fichier PipeTankToMotor.cpp.

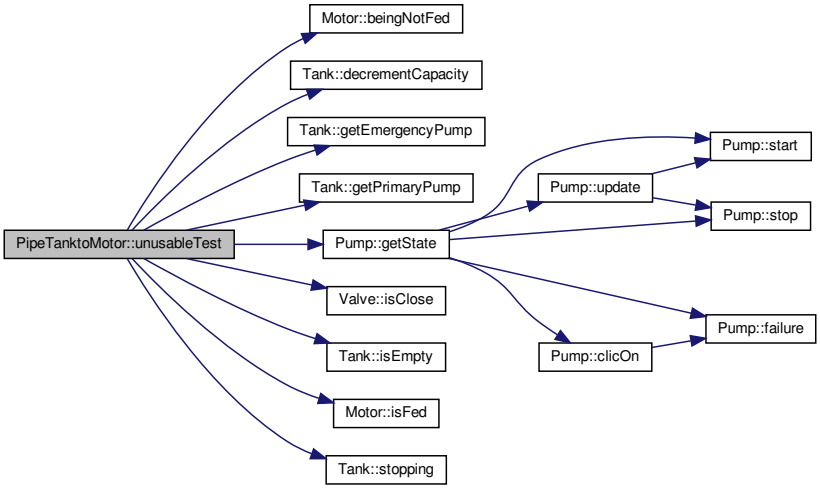
Références Motor : :beingNotFed(), Tank : :decrementCapacity(), Tank : :getEmergencyPump(), Tank : :get↩ PrimaryPump(), Pump : :getState(), Valve : :isClose(), Tank : :isEmpty(), Motor : :isFed(), Pipe : :m_Source, Pipe↩ : :m_Valve, et Tank : :stopping().

Référencé par run().

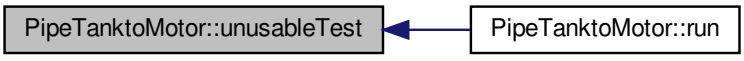
```
21 {
22     //Pompe principale alimente son moteur de base
23     if (!m_Source->isEmpty() && m_Source->getPrimaryPump().
      getState() == 0 && m_Source->getEmergencyPump().
      getState() != 0 && !m_Valve->isClose() && !m_Valve2->
      isClose())
24     {
25         m_Source->decrementCapacity(1);
26     }
27
28     //Vidange du moteur
29     else if (m_Source->isEmpty() && !m_Valve->isClose() && !m_Valve2->
      isClose() && m_Motor->isFed())
30     {
31         m_Motor->beingNotFed();
32     }
33
34     //Pompe principale en panne et pompe secondaire ne fonctionne pas
35     else if (!m_Source->isEmpty() && m_Source->
      getPrimaryPump().getState() == 2 && m_Source->
      getEmergencyPump().getState() != 0 && !m_Valve->
      isClose() && !m_Valve2->isClose() && m_Motor->isFed())
36     {
37         m_Source->stopping();
```

```
38          m_Motor->beingNotFed();
39      }
40
41      //Les 2 pompes pour le même moteur
42      else if (!m_Source->isEmpty() && m_Source->
        getPrimaryPump().getState() == 0 && m_Source->
        getEmergencyPump().getState() == 0 && !m_Valve->
        isClose() && !m_Valve2->isClose())
43      {
44          //Reduire la note (pompe secondaire ouverte pour rien)
45          m_Source->decrementCapacity(2);
46      }
47  }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.9.3.4 updateMotor()

```
void PipeTanktoMotor::updateMotor ( )
```

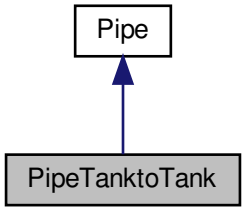
La documentation de cette classe a été générée à partir des fichiers suivants :

- include/PipeTankToMotor.h
- src/PipeTankToMotor.cpp

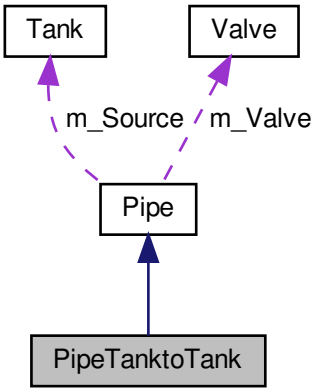
4.10 Référence de la classe PipeTanktoTank

`#include <PipeTankToTank.h>`

Graphe d'héritage de PipeTanktoTank :



Graphe de collaboration de PipeTanktoTank :



Fonctions membres publiques

- PipeTanktoTank (Tank ∗, Valve ∗, Tank ∗)
  Construct a new Pipe Tankto Tank : : Pipe Tankto Tank object.
- virtual void reconfiguration ()
  Cas custom.
- virtual void run ()
  Tank to Tank.

Membres hérités additionnels

4.10.1 Description détaillée

Définition à la ligne 5 du fichier PipeTankToTank.h.

4.10.2 Documentation des constructeurs et destructeur

4.10.2.1 PipeTanktoTank()

```
PipeTanktoTank::PipeTanktoTank (
            Tank * source,
            Valve * valve,
            Tank * desination )
```

Construct a new Pipe Tankto Tank : : Pipe Tankto Tank object.

Paramètres

source	
valve	
desination	

Définition à la ligne 11 du fichier PipeTankToTank.cpp.

```
11                                                                          :
      Pipe(source, valve), m_Destination(desination)
12 {
13 }
```

4.10.3 Documentation des fonctions membres

4.10.3.1 reconfiguration()

```
void PipeTanktoTank::reconfiguration ( )  [virtual]
```

Cas custom.

Implémente Pipe.

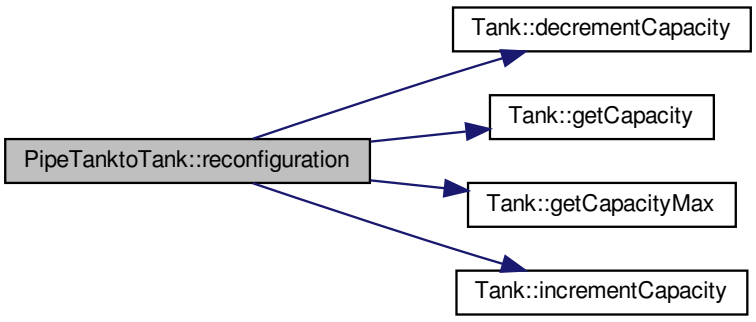
Définition à la ligne 19 du fichier PipeTankToTank.cpp.

Références Tank : :decrementCapacity(), Tank : :getCapacity(), Tank : :getCapacityMax(), Tank : :increment↩ Capacity(), et Pipe : :m_Source.

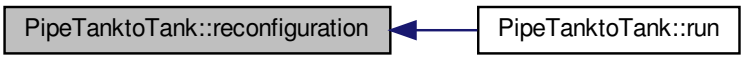
Référencé par run().

```
20 {
21
22     if (m_Source->getCapacity() < m_Destination->getCapacity())
23     {
24         if (m_Source->getCapacity() < m_Source->
      getCapacityMax() && m_Destination->getCapacity() > 0)
25         {
26             m_Source->incrementCapacity();
27             m_Destination->decrementCapacity(1);
28         }
29     }
30
31     else if (m_Source->getCapacity() > m_Destination->
      getCapacity())
32     {
33         if (m_Source->getCapacity() > 0 && m_Destination->
      getCapacity() < m_Destination->getCapacityMax())
34         {
35             m_Source->decrementCapacity(1);
36             m_Destination->incrementCapacity();
37         }
38     }
39 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.10.3.2 run()

```
void PipeTanktoTank::run ( )  [virtual]
```

Tank to Tank.

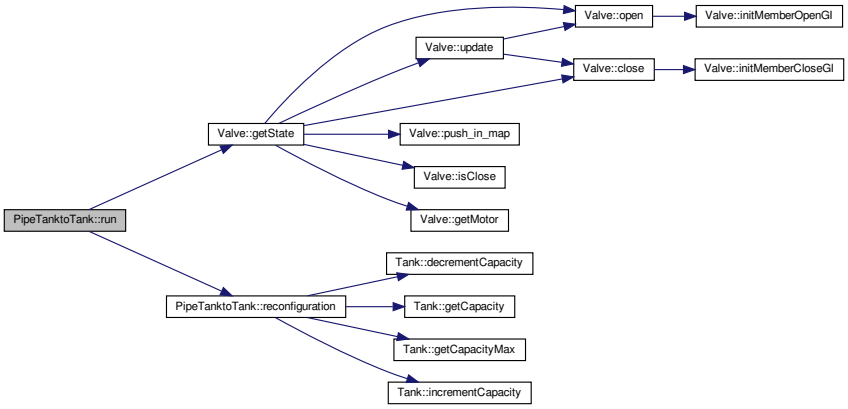
Réimplémentée à partir de Pipe.

Définition à la ligne 45 du fichier PipeTankToTank.cpp.

Références Valve : :getState(), Pipe : :m_Valve, et reconfiguration().

```
46 {
47     if (m_Valve->getState() == 1)
48     {
49         reconfiguration();
50     }
51 }
```

Voici le graphe d'appel pour cette fonction :



La documentation de cette classe a été générée à partir des fichiers suivants :
- include/PipeTankToTank.h
- src/PipeTankToTank.cpp

4.11 Référence de la classe Pump

`#include <Pump.h>`

Fonctions membres publiques

- void initMemberGl ()
  Représente les pompes (graphiquement)
- void drawPump ()
  Dessine une pompe.
- Vector2 getPosition ()
  Retourne la position d'une pompe.
- int getRadius ()
  Retourne le rayon d'une pompe.
- Pump ()
  Construct a new Pump : : Pump object.
- Pump (std : :string)
  Construct a new Pump : : Pump object.

- std : :string getName ()
  Retourne le nom de la pompe.
- StatePump getState ()
- void start ()
  Démarrage de la pompe.
- void stop ()
  Arrêt de la pompe.
- void failure ()
  Panne de la pompe.
- void clicOn ()
  Opération sur une pompe.
- void update ()
  Mise à jour d'une pompe.

4.11.1 Description détaillée

Définition à la ligne 7 du fichier Pump.h.

4.11.2 Documentation des constructeurs et destructeur

4.11.2.1 Pump() [1/2]

```
Pump::Pump ( )
```

Construct a new Pump : : Pump object.

Définition à la ligne 8 du fichier Pump.cpp.

```
9 {
10 }
```

4.11.2.2 Pump() [2/2]

```
Pump::Pump (
            std::string name )
```

Construct a new Pump : : Pump object.

Paramètres

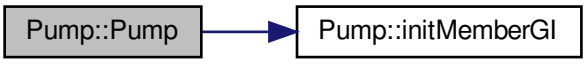
name	

Définition à la ligne 17 du fichier Pump.cpp.

Références initMemberGI().

```
17                          : m_Name(name), m_statePump(WORKING)
18 {
19     initMemberGI();
20 };
```

Voici le graphe d'appel pour cette fonction :



4.11.3 Documentation des fonctions membres

4.11.3.1 clicOn()

```
void Pump::clicOn ( )
```

Opération sur une pompe.

Définition à la ligne 134 du fichier Pump.cpp.

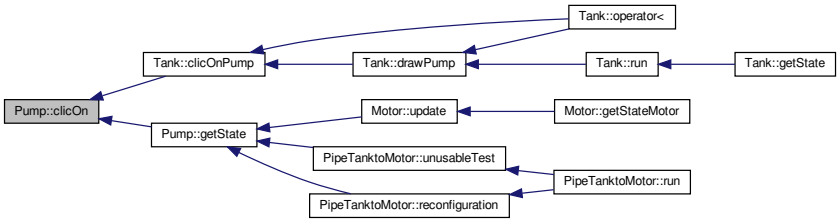
Références failure(), STOPPED, et WORKING.

Référencé par Tank : :clicOnPump(), et getState().

```
135 {
136     if (((m_statePump == STOPPED || m_statePump == WORKING) && IsMouseButtonReleased(
      MOUSE_LEFT_BUTTON)) && (GetMouseX() > (P_CENTERX - P_RADIUS) && GetMouseX() < (P_CENTERX + P_RADIUS)) && (GetMouseY(
      ) > (P_CENTERY - P_RADIUS) && GetMouseY() < (P_CENTERY + P_RADIUS)))
137     {
138         failure();
139     }
140 }
```

Voici le graphe d'appel pour cette fonction :

Voici le graphe des appelants de cette fonction :



4.11.3.2 drawPump()

```
void Pump::drawPump ( )
```

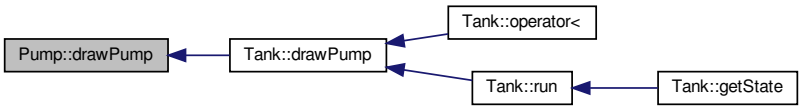
Dessine une pompe.

Définition à la ligne 125 du fichier Pump.cpp.

Référencé par Tank : :drawPump().

```
126 {
127     DrawCircle(P_CENTERX, P_CENTERY, P_RADIUS, P_COLOR);
128 }
```

Voici le graphe des appelants de cette fonction :



4.11.3.3 failure()

```
void Pump::failure ( )
```

Panne de la pompe.

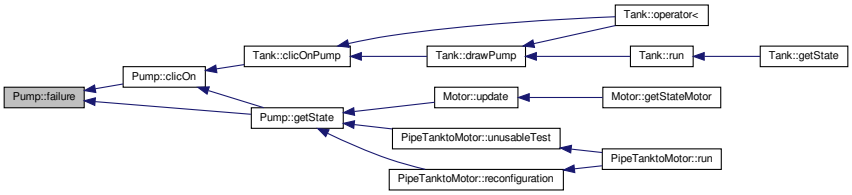
Définition à la ligne 56 du fichier Pump.cpp.

Références UNUSABLE.

Référencé par clicOn(), et getState().

```
57 {
58     m_statePump = UNUSABLE;
59     P_COLOR = PURPLE;
60 }
```

Voici le graphe des appelants de cette fonction :



4.11.3.4 getName()

```
std::string Pump::getName ( )
```

Retourne le nom de la pompe.

Renvoie

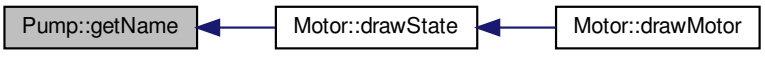
std : :string

Définition à la ligne 27 du fichier Pump.cpp.

Référencé par Motor : :drawState().

```
28 {
29     return m_Name;
30 }
```

Voici le graphe des appelants de cette fonction :



4.11.3.5 getPosition()

```
Vector2 Pump::getPosition ( )
```

Retourne la position d'une pompe.

Renvoie

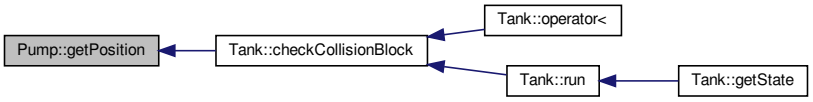
Vector2

Définition à la ligne 105 du fichier Pump.cpp.

Référencé par Tank : :checkCollisionBlock().

```
106 {
107     Vector2 v = {(float)P_CENTERX, (float)P_CENTERY};
108     return v;
109 }
```

Voici le graphe des appelants de cette fonction :



4.11.3.6 getRadius()

```
int Pump::getRadius ( )
```

Retourne le rayon d'une pompe.

Renvoie

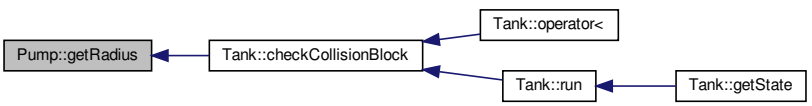
int

Définition à la ligne 116 du fichier Pump.cpp.

Référencé par Tank : :checkCollisionBlock().

```
117 {
118     return P_RADIUS;
119 }
```

Voici le graphe des appelants de cette fonction :



4.11.3.7 getState()

```
StatePump Pump::getState ( )  [inline]
```

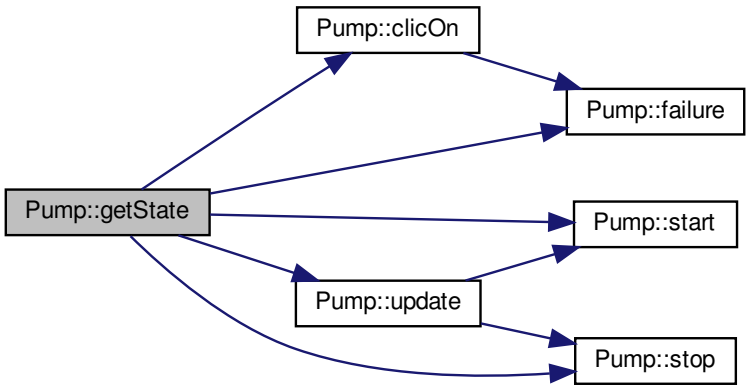
Définition à la ligne 37 du fichier Pump.h.

Références clicOn(), failure(), start(), stop(), et update().

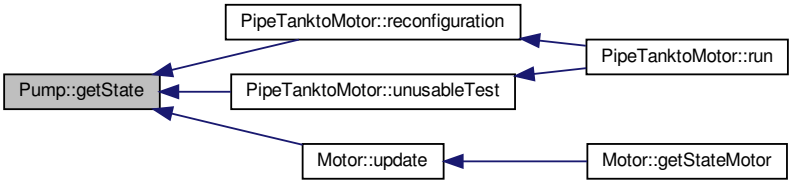
Référencé par PipeTanktoMotor : :reconfiguration(), PipeTanktoMotor : :unusableTest(), et Motor : :update().

```
37 { return m_statePump; }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.11.3.8 initMemberGl()

```
void Pump::initMemberGl ( )
```

Représente les pompes (graphiquement)

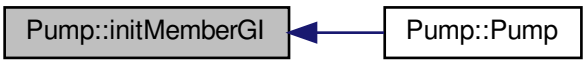
Définition à la ligne 66 du fichier Pump.cpp.

Références FL_HEIGHT, et FL_WIDTH.

Référencé par Pump().

```
67 {
68     if (m_Name.compare("P11") == 0)
69     {
70         P_CENTERX = (FL_WIDTH / 8) + ((FL_WIDTH / 6) / 3.5);
71         P_RADIUS = (FL_HEIGHT / 40);
72     }
73     else if (m_Name.compare("P12") == 0)
74     {
75         P_CENTERX = ((FL_WIDTH / 8) + (FL_WIDTH / 6)) - ((
      FL_WIDTH / 6) / 3.5);
76         P_RADIUS = (FL_HEIGHT / 40);
77     }
78     else if (m_Name.compare("P21") == 0)
79     {
80         P_CENTERX = FL_WIDTH * 0.47;
81         P_RADIUS = (FL_HEIGHT / 41.1764705882);
82     }
83     else if (m_Name.compare("P22") == 0)
84     {
85         P_CENTERX = FL_WIDTH * 0.53;
86         P_RADIUS = (FL_HEIGHT / 41.1764705882);
87     }
88     else if (m_Name.compare("P31") == 0)
89     {
90         P_CENTERX = (FL_WIDTH - ((FL_WIDTH / 8) + (FL_WIDTH / 6))) + ((
      FL_WIDTH / 6) / 3.5);
91         P_RADIUS = (FL_HEIGHT / 40);
92     }
93     else if (m_Name.compare("P32") == 0)
94     {
95         P_CENTERX = ((FL_WIDTH - ((FL_WIDTH / 8) + (FL_WIDTH / 6))) + (
      FL_WIDTH / 6)) - ((FL_WIDTH / 6) / 3.5);
96         P_RADIUS = (FL_HEIGHT / 40);
97     }
98 }
```

Voici le graphe des appelants de cette fonction :



4.11.3.9 start()

```
void Pump::start ( )
```

Démarrage de la pompe.

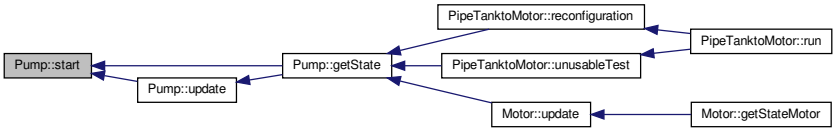
Définition à la ligne 36 du fichier Pump.cpp.

Références WORKING.

Référencé par getState(), et update().

```
37 {
38     m_statePump = WORKING;
39     P_COLOR = BLACK;
40 }
```

Voici le graphe des appelants de cette fonction :



4.11.3.10 stop()

```
void Pump::stop ( )
```

Arrêt de la pompe.

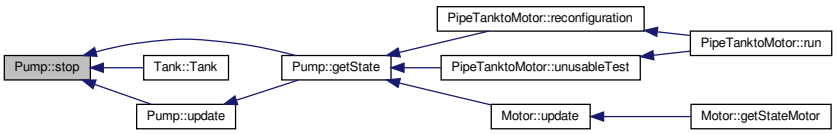
Définition à la ligne 46 du fichier Pump.cpp.

Références STOPPED.

Référencé par getState(), Tank : :Tank(), et update().

```
47 {
48     m_statePump = STOPPED;
49     P_COLOR = RED;
50 }
```

Voici le graphe des appelants de cette fonction :



4.11.3.11 update()

```
void Pump::update ( )
```

Mise à jour d'une pompe.

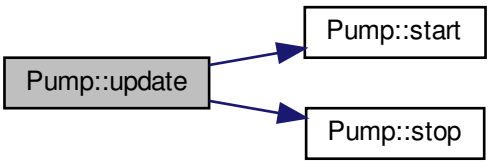
Définition à la ligne 146 du fichier Pump.cpp.

Références start(), stop(), STOPPED, et WORKING.

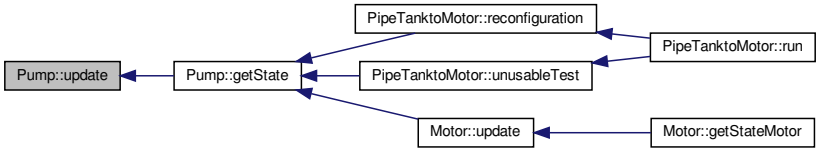
Référencé par getState().

```
147 {
148     if (m_statePump == STOPPED)
149     {
150         start();
151     }
152
153     else if (m_statePump == WORKING)
154     {
155         stop();
156     }
157 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



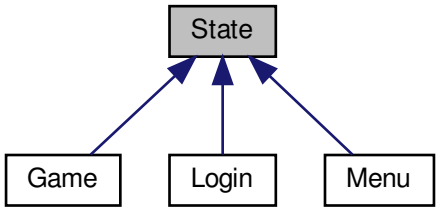
La documentation de cette classe a été générée à partir des fichiers suivants :

- include/Pump.h
- src/Pump.cpp

4.12 Référence de la classe State

`#include <State.h>`

Graphe d'héritage de State :



Fonctions membres publiques

- State (float, float, char ∗)
  Construct a new State : : State object.
- virtual void run (App ∗app)=0
- virtual ∼State ()
  Destroy the State : : State object.

Attributs publics

- const float W_WIDTH
- const float W_HEIGHT
- const char ∗ W_TITLE

4.12.1 Description détaillée

Définition à la ligne 6 du fichier State.h.

4.12.2 Documentation des constructeurs et destructeur

4.12.2.1 State()

```
State::State (
            float whidth,
            float height,
            char * title )
```

Construct a new State : : State object.

Paramètres

whidth	
height	
title	

Définition à la ligne 10 du fichier State.cpp.

```
10                                                  : W_WIDTH(whidth),
      W_HEIGHT(height), W_TITLE(title)
11 {
12 }
```

4.12.2.2 ~State()

```
State::~State ( )  [virtual]
```

Destroy the State : : State object.

Définition à la ligne 18 du fichier State.cpp.

```
19 {
20 }
```

4.12.3 Documentation des fonctions membres

4.12.3.1 run()

```
virtual void State::run (
             App * app )  [pure virtual]
```

Implémenté dans Login, Game, et Menu.

4.12.4 Documentation des données membres

4.12.4.1 W_HEIGHT

```
const float State::W_HEIGHT
```

Définition à la ligne 12 du fichier State.h.

Référencé par Game : :Game(), Menu : :getBackBlock(), Game : :getBackBlock(), Login : :getBackBlock(), Menu↩ : :getStartBlock(), Menu : :Menu(), et Game : :run().

4.12.4.2 W_TITLE

```
const char* State::W_TITLE
```

Définition à la ligne 13 du fichier State.h.

Référencé par Game : :Game(), et Menu : :Menu().

4.12.4.3 W_WIDTH

```
const float State::W_WIDTH
```

Définition à la ligne 11 du fichier State.h.

Référencé par Login : :drawEnterBlock(), Login : :drawIdBlock(), Login : :drawPasswordBlock(), Game : :Game(), Menu : :getBackBlock(), Game : :getBackBlock(), Login : :getBackBlock(), Login : :getEnterBlock(), Login : :getId↩ Block(), Login : :getPasswordBlock(), Menu : :getStartBlock(), Menu : :Menu(), et Game : :run().

La documentation de cette classe a été générée à partir des fichiers suivants :
- include/State.h
- src/State.cpp

4.13 Référence de la classe Tank

```
#include <Tank.h>
```

Fonctions membres publiques

- Tank (std : :string, int, std : :string, std : :string)
  Construct a new Tank : : Tank object.
- float getCapacity ()
  Retourne la capacité d'un réservoir.
- float getCapacityMax ()
  Retourne la capacité max d'un réservoir.
- StateTank getState ()
- Pump & getPrimaryPump ()
  Retourne la pompe primaire d'un réservoir.
- Pump & getEmergencyPump ()
  Retourne la pompe de secours d'un réservoir.
- void incrementCapacity ()
  Augmente la capacité d'un réservoir.
- void decrementCapacity (int)
  Baisse la capacité d'un réservoir.
- void flowing ()
  Écoulement d'un réservoir.
- void emptying ()
  Réservoir à sec.
- void stopping ()
  Réservoir à l'arrêt.
- bool isEmpty ()
  Connaître l'état d'un réservoir.
- void run ()
  Init un Réservoir.
- bool operator< (const Tank &other) const
- void initMemberGl ()

Init graphique.
- void drawTank ()
  Dessine les réservoirs.
- void drawFlowingTank ()
  Dessines les écoulements.
- void drawPump ()
  Dessine les pompes du réservoir.
- Rectangle getTankBlock ()
  Représentation d'un réservoir.
- Rectangle getFlowingTankBlock ()
  Retourne les réservoirs en écoulements.
- void clicOnPump ()
  Clique sur une pompe.
- void checkCollisionBlock ()
  Vérifie les cliques de la souris.

4.13.1 Description détaillée

Définition à la ligne 10 du fichier Tank.h.

4.13.2 Documentation des constructeurs et destructeur

4.13.2.1 Tank()

```
Tank::Tank (
            std::string name,
            int capacity,
            std::string namePPump,
            std::string nameEPump )
```

Construct a new Tank : : Tank object.

Paramètres

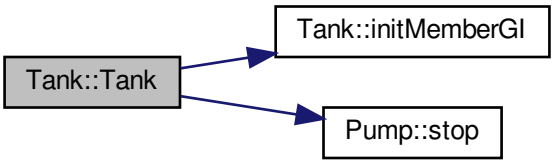
name	
capacity	
namePPump	
nameEPump	

Définition à la ligne 35 du fichier Tank.cpp.

Références initMemberGI(), et Pump : :stop().

```
35                                                                          : m_Name(name), m_capacity
    (capacity), m_stateTank(FULL), m_primaryPump(namePPump), m_emergencyPump(nameEPump)
36 {
37     m_capacity_Max = m_capacity;
38     m_emergencyPump.stop();
39     initMemberGI();
40 }
```

Voici le graphe d'appel pour cette fonction :



4.13.3 Documentation des fonctions membres

4.13.3.1 checkCollisionBlock()

```
void Tank::checkCollisionBlock ( )
```

Vérifie les cliques de la souris.

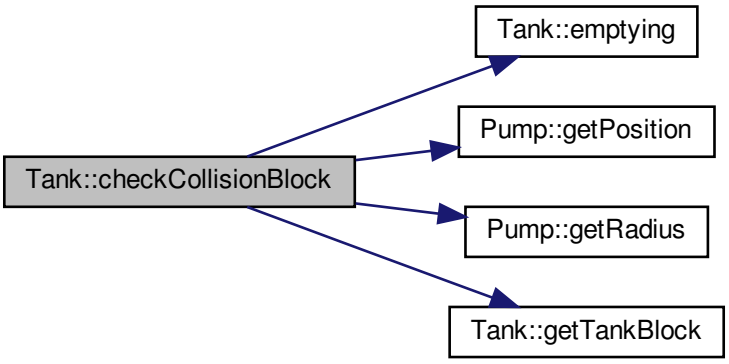
Définition à la ligne 264 du fichier Tank.cpp.

Références emptying(), FLOW, Pump : :getPosition(), Pump : :getRadius(), et getTankBlock().

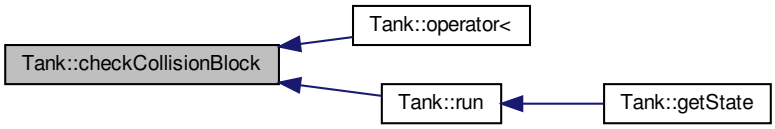
Référencé par operator<(), et run().

```
265 {
266     if (CheckCollisionPointRec(GetMousePosition(), getTankBlock()))
267     {
268         mouseOnTank = true;
269 
270         if (IsMouseButtonReleased(MOUSE_LEFT_BUTTON))
271         {
272             clicOnTank = true;
273         }
274     }
275     else
276         mouseOnTank = false;
277 
278     if (CheckCollisionPointRec(GetMousePosition(), getTankBlock()) && !
      CheckCollisionPointCircle(GetMousePosition(), m_primaryPump.getPosition(), m_primaryPump.
      getRadius()) && !CheckCollisionPointCircle(GetMousePosition(), m_emergencyPump.
      getPosition(), m_emergencyPump.getRadius()) && IsMouseButtonReleased(MOUSE_LEFT_BUTTON)
       /*&& m_stateTank != FLOW*/)
279     {
280         m_stateTank = FLOW;
281         emptying();
282     }
283 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.13.3.2 clicOnPump()

```
void Tank::clicOnPump ( )
```

Clique sur une pompe.

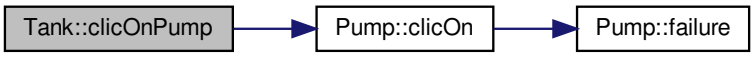
Définition à la ligne 243 du fichier Tank.cpp.

Références Pump : :clicOn().

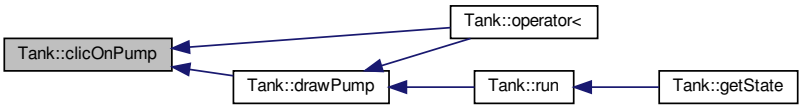
Référencé par drawPump(), et operator<().

```
244 {
245     m_primaryPump.clicOn();
246     m_emergencyPump.clicOn();
247 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.13.3.3 decrementCapacity()

```
void Tank::decrementCapacity (
            int mult )
```

Baisse la capacité d'un réservoir.

Paramètres

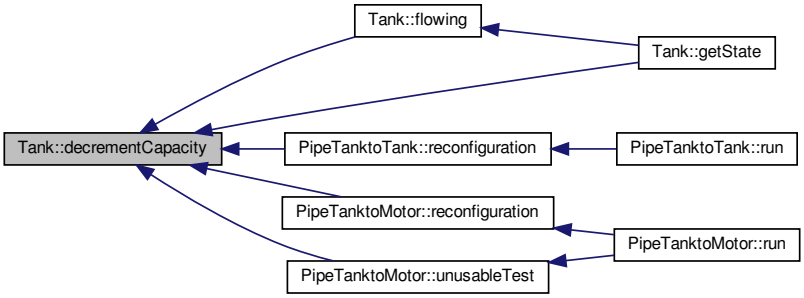
mult	

Définition à la ligne 96 du fichier Tank.cpp.

Référencé par flowing(), getState(), PipeTanktoTank : :reconfiguration(), PipeTanktoMotor : :reconfiguration(), et PipeTanktoMotor : :unusableTest().

```
97 {
98     m_capacity -= 0.1 * mult;
99 }
```

Voici le graphe des appelants de cette fonction :



4.13.3.4 drawFlowingTank()

```
void Tank::drawFlowingTank ( )
```

Dessines les écoulements.

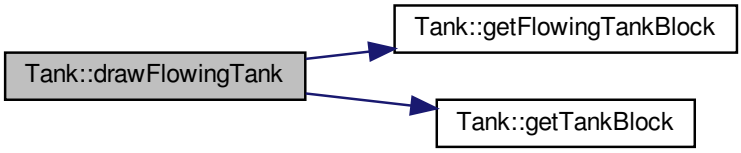
Définition à la ligne 192 du fichier Tank.cpp.

Références getFlowingTankBlock(), et getTankBlock().

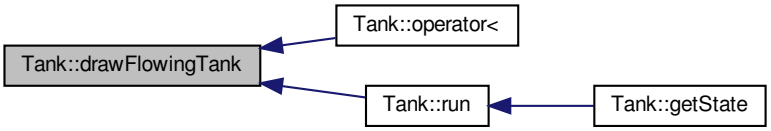
Référencé par operator<(), et run().

```
193 {
194     Rectangle tank = getTankBlock();
195     Rectangle flowingTank = getFlowingTankBlock();
196 
197     DrawText(m_Name.c_str(), tank.x + tank.width / 6, tank.y + tank.height * 0.25, tank.width / 4.5, GRAY);
198     DrawRectangleRec(flowingTank, (Color){T_COLOR.r - 50, T_COLOR.g - 50, T_COLOR.b - 50, T_COLOR.a - 70});
199 
200     if (mouseOnTank)
201         DrawRectangleLinesEx(tank, 4, BLUE);
202 
203     else
204         DrawRectangleLinesEx(tank, 4, DARKGRAY);
205 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.13.3.5 drawPump()

```
void Tank::drawPump ( )
```

Dessine les pompes du réservoir.

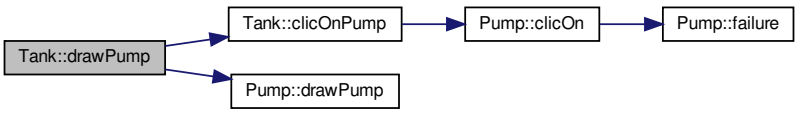
Définition à la ligne 253 du fichier Tank.cpp.

Références clicOnPump(), et Pump : :drawPump().

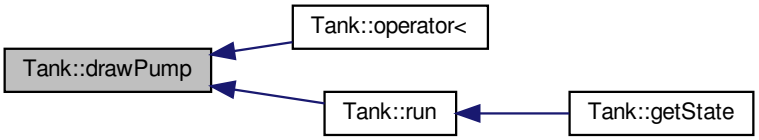
Référencé par operator<(), et run().

```
254 {
255     clicOnPump();
256     m_primaryPump.drawPump();
257     m_emergencyPump.drawPump();
258 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.13.3.6 drawTank()

```
void Tank::drawTank ( )
```

Dessine les réservoirs.

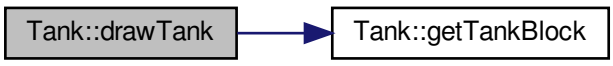
Définition à la ligne 175 du fichier Tank.cpp.

Références getTankBlock().

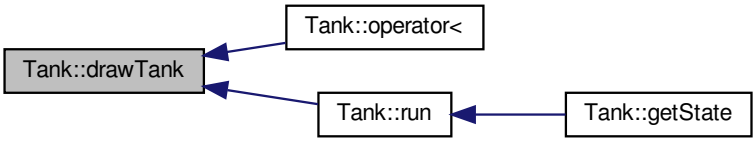
Référencé par operator<(), et run().

```
176 {
177     Rectangle tank = getTankBlock();
178 
179     DrawRectangleRec(tank, T_COLOR);
180 
181     if (mouseOnTank)
182         DrawRectangleLinesEx(tank, 2, BLUE);
183 
184     else
185         DrawRectangleLinesEx(tank, 4, DARKGRAY);
186 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.13.3.7 emptying()

```
void Tank::emptying ( )
```

Réservoir à sec.

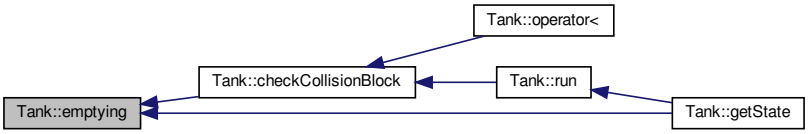
Définition à la ligne 115 du fichier Tank.cpp.

Références EMPTY.

Référencé par checkCollisionBlock(), et getState().

```
116 {
117     m_capacity = 0;
118     m_stateTank = EMPTY;
119 }
```

Voici le graphe des appelants de cette fonction :



4.13.3.8 flowing()

```
void Tank::flowing ( )
```

Écoulement d'un réservoir.

Définition à la ligne 105 du fichier Tank.cpp.

Références decrementCapacity(), et FLOW.

Référencé par getState().

```
106 {
107     m_stateTank = FLOW;
108     decrementCapacity(1);
109 }
```

Voici le graphe d'appel pour cette fonction :

Voici le graphe des appelants de cette fonction :

4.13.3.9 getCapacity()

```
float Tank::getCapacity ( )
```

Retourne la capacité d'un réservoir.

Renvoie

float

Définition à la ligne 47 du fichier Tank.cpp.

Référencé par PipeTanktoTank : :reconfiguration().

```
48 {
49     return m_capacity;
50 }
```

Voici le graphe des appelants de cette fonction :

4.13.3.10 getCapacityMax()

```
float Tank::getCapacityMax ( )
```

Retourne la capacité max d'un réservoir.

Renvoie

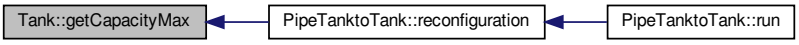
float

Définition à la ligne 57 du fichier Tank.cpp.

Référencé par PipeTanktoTank : :reconfiguration().

```
58 {
59     return m_capacity_Max;
60 }
```

Voici le graphe des appelants de cette fonction :



4.13.3.11 getEmergencyPump()

```
Pump & Tank::getEmergencyPump ( )
```

Retourne la pompe de secours d'un réservoir.

Renvoie

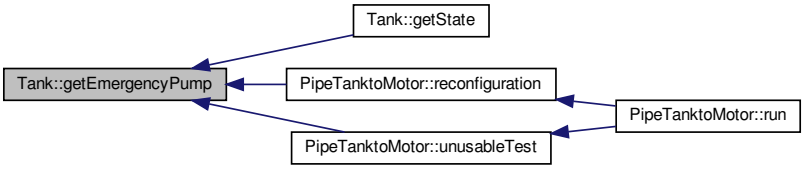
Pump&

Définition à la ligne 77 du fichier Tank.cpp.

Référencé par getState(), PipeTanktoMotor : :reconfiguration(), et PipeTanktoMotor : :unusableTest().

```
78 {
79     return m_emergencyPump;
80 }
```

Voici le graphe des appelants de cette fonction :



4.13.3.12 getFlowingTankBlock()

```
Rectangle Tank::getFlowingTankBlock ( )
```

Retourne les réservoirs en écoulements.

Renvoie

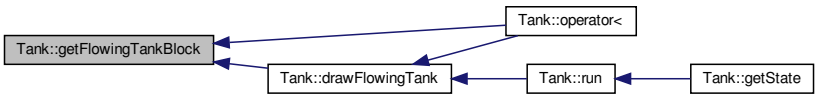
Rectangle

Définition à la ligne 223 du fichier Tank.cpp.

Référencé par drawFlowingTank(), et operator<().

```
224 {
225     Rectangle t_rec;
226 
227     if (m_Name.compare("Tank1") == 0 || m_Name.compare("Tank3") == 0)
228     {
229         t_rec = {T_POSX + 3, (T_POSY + 3) + (200 - m_capacity), T_WIDTH - 3, (T_HEIGHT - 3) - (200 -
    m_capacity)};
230     }
231     else
232     {
233         t_rec = {T_POSX + 3, (T_POSY + 3) + (200 - m_capacity * 2), T_WIDTH - 3, (T_HEIGHT - 3) - (200 -
    m_capacity * 2)};
234     }
235 
236     return t_rec;
237 }
```

Voici le graphe des appelants de cette fonction :



4.13.3.13 getPrimaryPump()

```
Pump & Tank::getPrimaryPump ( )
```

Retourne la pompe primaire d'un réservoir.

Renvoie

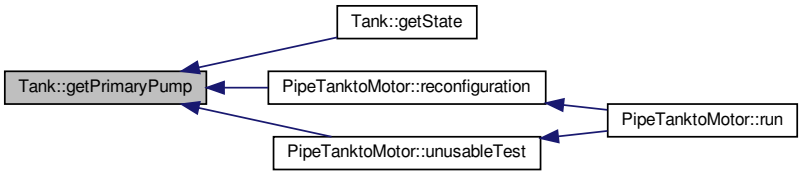
Pump&

Définition à la ligne 67 du fichier Tank.cpp.

Référencé par getState(), PipeTanktoMotor : :reconfiguration(), et PipeTanktoMotor : :unusableTest().

```
68 {
69     return m_primaryPump;
70 }
```

Voici le graphe des appelants de cette fonction :



4.13.3.14 getState()

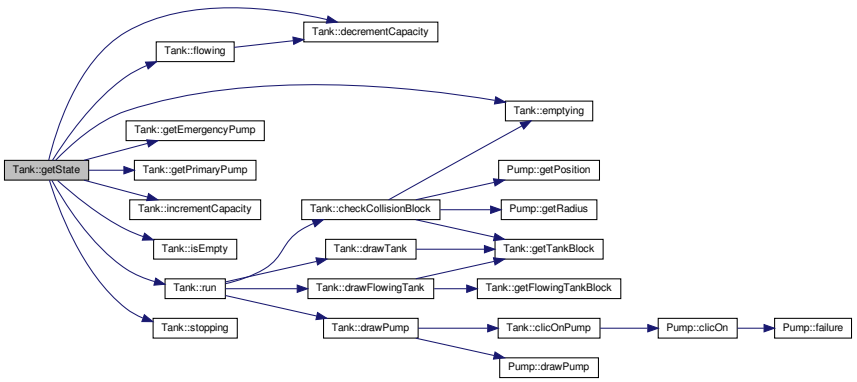
```
StateTank Tank::getState ( )  [inline]
```

Définition à la ligne 37 du fichier Tank.h.

Références decrementCapacity(), emptying(), flowing(), getEmergencyPump(), getPrimaryPump(), increment↩ Capacity(), isEmpty(), run(), et stopping().

```
37 {return m_stateTank;}
```

Voici le graphe d'appel pour cette fonction :



4.13.3.15 getTankBlock()

```
Rectangle Tank::getTankBlock ( )
```

Représentation d'un réservoir.

Renvoie

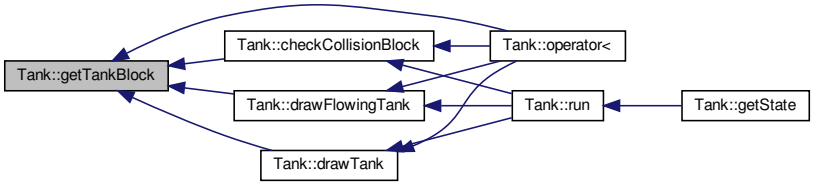
Rectangle

Définition à la ligne 212 du fichier Tank.cpp.

Référencé par checkCollisionBlock(), drawFlowingTank(), drawTank(), et operator<().

```
213 {
214     Rectangle t_rec = {T_POSX, T_POSY, T_WIDTH, T_HEIGHT};
215     return t_rec;
216 }
```

Voici le graphe des appelants de cette fonction :



4.13.3.16 incrementCapacity()

```
void Tank::incrementCapacity ( )
```

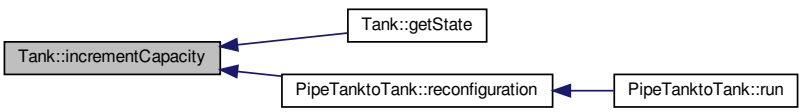
Augmente la capacité d'un réservoir.

Définition à la ligne 86 du fichier Tank.cpp.

Référencé par getState(), et PipeTanktoTank : :reconfiguration().

```
87 {
88     m_capacity += 0.1;
89 }
```

Voici le graphe des appelants de cette fonction :



4.13.3.17 initMemberGl()

```
void Tank::initMemberGl ( )
```

Init graphique.

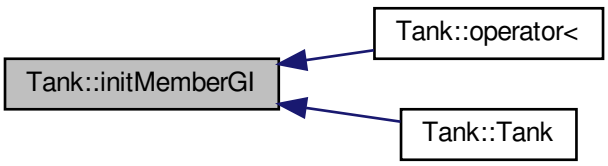
Définition à la ligne 147 du fichier Tank.cpp.

Références FL_WIDTH.

Référencé par operator<(), et Tank().

```
148 {
149     if (m_Name.compare("Tank1") == 0)
150     {
151         T_WIDTH = (FL_WIDTH / 6);
152         T_POSX = (FL_WIDTH / 8);
153         T_COLOR = (Color){255, 161, 150, 255};
154     }
155 
156     else if (m_Name.compare("Tank2") == 0)
157     {
158         T_WIDTH = (FL_WIDTH / 8);
159         T_POSX = ((FL_WIDTH / 2) - (T_WIDTH / 2));
160         T_COLOR = (Color){0, 228, 100, 255};
161     }
162 
163     else if (m_Name.compare("Tank3") == 0)
164     {
165         T_WIDTH = (FL_WIDTH / 6);
166         T_POSX = (FL_WIDTH - ((FL_WIDTH / 8) + T_WIDTH));
167         T_COLOR = (Color){102, 191, 255, 255};
168     }
169 }
```

Voici le graphe des appelants de cette fonction :



4.13.3.18 isEmpty()

```
bool Tank::isEmpty ( )
```

Connaître l'état d'un réservoir.

Renvoie

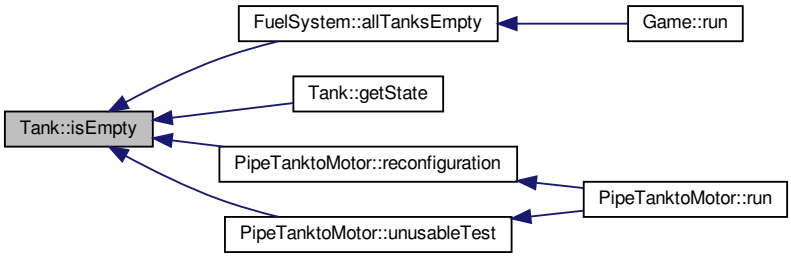
true
false

Définition à la ligne 136 du fichier Tank.cpp.

Référencé par FuelSystem : :allTanksEmpty(), getState(), PipeTanktoMotor : :reconfiguration(), et PipeTankto↩ Motor : :unusableTest().

```
137 {
138     if (m_capacity <= 0)
139         return true;
140     return false;
141 }
```

Voici le graphe des appelants de cette fonction :



4.13.3.19 operator<()

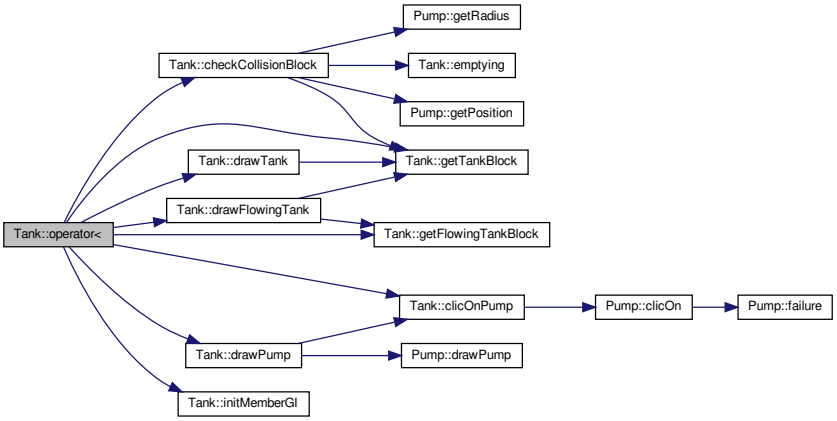
```
bool Tank::operator< (
            const Tank & other ) const  [inline]
```

Définition à la ligne 48 du fichier Tank.h.

Références checkCollisionBlock(), clicOnPump(), drawFlowingTank(), drawPump(), drawTank(), getFlowingTank↩ Block(), getTankBlock(), et initMemberGI().

```
48 { return m_Name < other.m_Name; }
```

Voici le graphe d'appel pour cette fonction :



4.13.3.20 run()

```
void Tank::run ( )
```

Init un Réservoir.

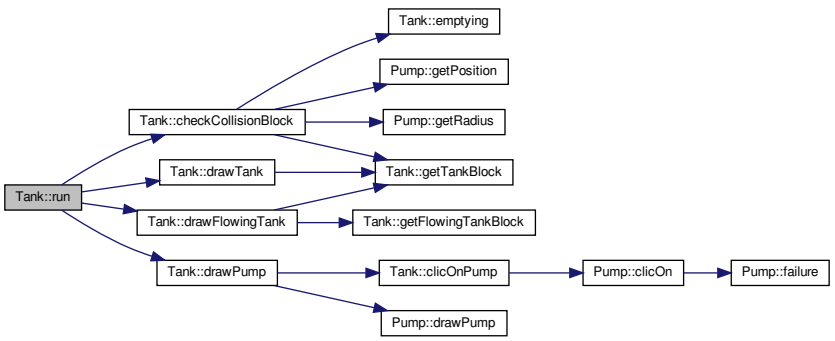
Définition à la ligne 289 du fichier Tank.cpp.

Références checkCollisionBlock(), drawFlowingTank(), drawPump(), et drawTank().

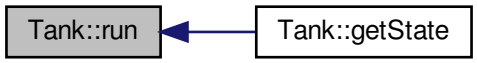
Référencé par getState().

```
290 {
291     checkCollisionBlock();
292
293     drawTank();
294     drawFlowingTank();
295     drawPump();
296 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.13.3.21 stopping()

```
void Tank::stopping ( )
```

Réservoir à l'arrêt.

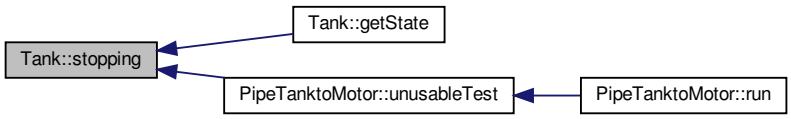
Définition à la ligne 125 du fichier Tank.cpp.

Références STOP.

Référencé par getState(), et PipeTanktoMotor : :unusableTest().

```
126 {
127     m_stateTank = STOP;
128 }
```

Voici le graphe des appelants de cette fonction :



La documentation de cette classe a été générée à partir des fichiers suivants :
- include/Tank.h
- src/Tank.cpp

4.14 Référence de la classe Valve

```
#include <Valve.h>
```

Fonctions membres publiques

- void initMemberCloseGI ()
  Représentation Vanne Fermer (graphiquement)
- void initMemberOpenGI ()
  Representation Vanne Ouvert (graphiquement)
- void drawValve ()
  Dessine Vanne.
- Valve (std : :string)
  Construct a new Valve : : Valve object.
- std : :string getName ()
  Retourne le nom de la Vanne.
- StateValve getState ()
- void open ()
  Ouvre la vanne.
- void close ()
  Ferme la Vanne.
- void update ()
  Met à jour la Vanne.
- void push_in_map (Tank ∗, Motor ∗)
  Réservoir to Vanne to Moteur.
- bool isClose ()
  Connaitre l'état de la vanne.
- Motor ∗ getMotor (Tank ∗)
  Retourne le moteur lié à la vanne.

4.14.1 Description détaillée

Définition à la ligne 12 du fichier Valve.h.

4.14.2 Documentation des constructeurs et destructeur

4.14.2.1 Valve()

```
Valve::Valve (
            std::string valveName )
```

Construct a new Valve : : Valve object.

Paramètres

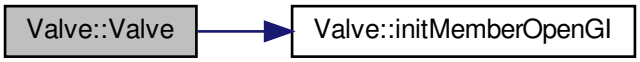
valveName	

Définition à la ligne 9 du fichier Valve.cpp.

Références initMemberOpenGI().

```
9                               : m_stateValve(OPEN), m_valveName(valveName)
10 {
11     initMemberOpenGI();
12 };
```

Voici le graphe d'appel pour cette fonction :



4.14.3 Documentation des fonctions membres

4.14.3.1 close()

```
void Valve::close ( )
```

Ferme la Vanne.

Définition à la ligne 38 du fichier Valve.cpp.

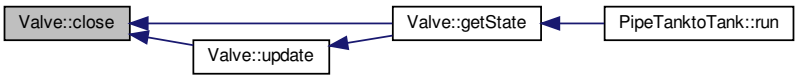
Références initMemberCloseGI().

Référencé par getState(), et update().

```
39 {
40     initMemberCloseGI();
41     m_stateValve = CLOSE;
42 }
```

Voici le graphe d'appel pour cette fonction :

Voici le graphe des appelants de cette fonction :



4.14.3.2 drawValve()

```
void Valve::drawValve ( )
```

Dessine Vanne.

Définition à la ligne 164 du fichier Valve.cpp.

```
165 {
166     DrawCircle(V_CENTERX, V_CENTERY, V_RADIUS, V_COLOR);
167     DrawRectangle(VR_POSX, VR_POSY, VR_WIDTH, VR_HEIGHT, VR_COLOR);
168 }
```

4.14.3.3 getMotor()

```
Motor * Valve::getMotor (
            Tank * tank )
```

Retourne le moteur lié à la vanne.

Paramètres

tank	

Renvoie

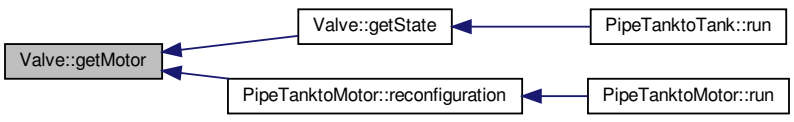
Motor∗

Définition à la ligne 200 du fichier Valve.cpp.

Référencé par getState(), et PipeTanktoMotor : :reconfiguration().

```
201 {
202     return map_Valve[tank];
203 }
```

Voici le graphe des appelants de cette fonction :



4.14.3.4 getName()

```
std::string Valve::getName ( )
```

Retourne le nom de la Vanne.

Renvoie

std : :string

Définition à la ligne 19 du fichier Valve.cpp.

```
20 {
21     return m_valveName;
22 }
```

4.14.3.5 getState()

```
StateValve Valve::getState ( )  [inline]
```

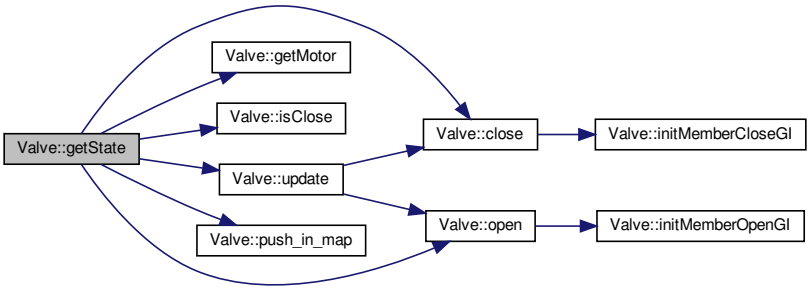
Définition à la ligne 43 du fichier Valve.h.

Références close(), getMotor(), isClose(), open(), push_in_map(), et update().

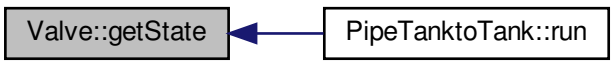
Référencé par PipeTanktoTank : :run().

```
43 { return m_stateValve; }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :



4.14.3.6 initMemberCloseGI()

```
void Valve::initMemberCloseGI ( )
```

Représentation Vanne Fermer (graphiquement)

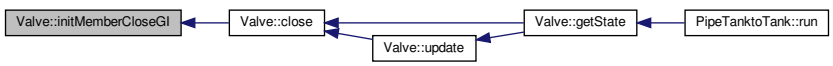
Définition à la ligne 113 du fichier Valve.cpp.

Références FL_HEIGHT, et FL_WIDTH.

Référencé par close().

```
114 {
115     VR_COLOR = (Color){232, 140, 8, 255};
116     VR_WIDTH = ((FL_HEIGHT / 40) * 2);
117     VR_HEIGHT = ((FL_HEIGHT / 40) / 2);
118 
119     if (m_valveName.compare("VT12") == 0)
120     {
121         V_CENTERX = (((FL_WIDTH / 8) + (FL_WIDTH / 6)) + (((
      FL_WIDTH / 2) - ((FL_WIDTH / 8) / 2)))) / 2;
122         V_CENTERY = (FL_HEIGHT / 50) + ((FL_HEIGHT / 5) / 2);
123         VR_POSX = V_CENTERX - V_RADIUS;
124         VR_POSY = V_CENTERY - VR_HEIGHT / 2;
125     }
126 
127     else if (m_valveName.compare("VT23") == 0)
128     {
129         V_CENTERX = ((((FL_WIDTH / 2) - ((FL_WIDTH / 8) / 2)) + (
      FL_WIDTH / 8)) + ((FL_WIDTH - ((FL_WIDTH / 8) + (
      FL_WIDTH / 6))))) / 2;
130         V_CENTERY = (FL_HEIGHT / 50) + ((FL_HEIGHT / 5) / 2);
131         VR_POSX = (((((FL_WIDTH / 2) - ((FL_WIDTH / 8) / 2)) + (
      FL_WIDTH / 8)) + ((FL_WIDTH - ((FL_WIDTH / 8) + (
      FL_WIDTH / 6))))) / 2) - (VR_WIDTH / 2);
132         VR_POSY = ((FL_HEIGHT / 50) + (FL_HEIGHT / 5) / 2) - (VR_HEIGHT / 2);
133     }
134 
135     else if (m_valveName.compare("V12") == 0)
136     {
137         V_CENTERX = (((FL_WIDTH / 8) + (FL_WIDTH / 6)) + (((
      FL_WIDTH / 2) - ((FL_WIDTH / 8) / 2)))) / 2;
138         V_CENTERY = (FL_HEIGHT - (FL_HEIGHT / 2.5)) * 0.8;
139         VR_POSX = ((((FL_WIDTH / 8) + (FL_WIDTH / 6)) + (((
      FL_WIDTH / 2) - ((FL_WIDTH / 8) / 2)))) / 2) - (VR_WIDTH / 2);
140         VR_POSY = (FL_HEIGHT - (FL_HEIGHT / 2.5)) * 0.8 - (VR_HEIGHT / 2);
141     }
142 
143     else if (m_valveName.compare("V13") == 0)
144     {
145         V_CENTERX = ((((FL_WIDTH / 2) - ((FL_WIDTH / 8) / 2)) + (
      FL_WIDTH / 8)) + ((FL_WIDTH - ((FL_WIDTH / 8) + (
      FL_WIDTH / 6))))) / 2;
146         V_CENTERY = (FL_HEIGHT - (FL_HEIGHT / 2.5)) * 0.6;
147         VR_POSX = (((((FL_WIDTH / 2) - ((FL_WIDTH / 8) / 2)) + (
      FL_WIDTH / 8)) + ((FL_WIDTH - ((FL_WIDTH / 8) + (
      FL_WIDTH / 6))))) / 2) - (VR_WIDTH / 2);
148         VR_POSY = (FL_HEIGHT - (FL_HEIGHT / 2.5)) * 0.6 - (VR_HEIGHT / 2);
149     }
150 
151     else if (m_valveName.compare("V23") == 0)
152     {
153         V_CENTERX = ((((FL_WIDTH / 2) - ((FL_WIDTH / 8) / 2)) + (
      FL_WIDTH / 8)) + ((FL_WIDTH - ((FL_WIDTH / 8) + (
      FL_WIDTH / 6))))) / 2;
154         V_CENTERY = (FL_HEIGHT - (FL_HEIGHT / 2.5)) * 0.9;
155         VR_POSX = (((((FL_WIDTH / 2) - ((FL_WIDTH / 8) / 2)) + (
      FL_WIDTH / 8)) + ((FL_WIDTH - ((FL_WIDTH / 8) + (
      FL_WIDTH / 6))))) / 2) - (VR_WIDTH / 2);
156         VR_POSY = (FL_HEIGHT - (FL_HEIGHT / 2.5)) * 0.9 - (VR_HEIGHT / 2);
157     }
158 }
```

Voici le graphe des appelants de cette fonction :



4.14.3.7 initMemberOpenGl()

```
void Valve::initMemberOpenGl ( )
```

Representation Vanne Ouvert (graphiquement)

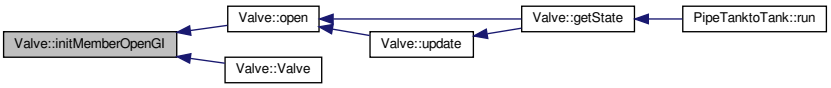
Définition à la ligne 62 du fichier Valve.cpp.

Références FL_HEIGHT, et FL_WIDTH.

Référencé par open(), et Valve().

```
63 {
64     VR_COLOR = LIGHTGRAY;
65     VR_WIDTH = ((FL_HEIGHT / 40) / 2);
66     VR_HEIGHT = ((FL_HEIGHT / 40) * 2);
67
68     if (m_valveName.compare("VT12") == 0)
69     {
70         V_CENTERX = (((FL_WIDTH / 8) + (FL_WIDTH / 6)) + (((
      FL_WIDTH / 2) - ((FL_WIDTH / 8) / 2)))) / 2;
71         V_CENTERY = (FL_HEIGHT / 50) + ((FL_HEIGHT / 5) / 2);
72         VR_POSX = (((((FL_WIDTH / 8) + (FL_WIDTH / 6)) + (((
      FL_WIDTH / 2) - ((FL_WIDTH / 8) / 2)))) / 2) - (VR_WIDTH / 2);
73         VR_POSY = ((FL_HEIGHT / 50) + (FL_HEIGHT / 5) / 2) - (VR_HEIGHT / 2);
74     }
75
76     else if (m_valveName.compare("VT23") == 0)
77     {
78         V_CENTERX = ((((FL_WIDTH / 2) - ((FL_WIDTH / 8) / 2)) + (
      FL_WIDTH / 8)) + ((FL_WIDTH - ((FL_WIDTH / 8) + (
      FL_WIDTH / 6))))) / 2;
79         V_CENTERY = (FL_HEIGHT / 50) + ((FL_HEIGHT / 5) / 2);
80         VR_POSX = (((((FL_WIDTH / 2) - ((FL_WIDTH / 8) / 2)) + (
      FL_WIDTH / 8)) + ((FL_WIDTH - ((FL_WIDTH / 8) + (
      FL_WIDTH / 6))))) / 2) - (VR_WIDTH / 2);
81         VR_POSY = ((FL_HEIGHT / 50) + (FL_HEIGHT / 5) / 2) - (VR_HEIGHT / 2);
82     }
83
84     else if (m_valveName.compare("V12") == 0)
85     {
86         V_CENTERX = (((FL_WIDTH / 8) + (FL_WIDTH / 6)) + (((
      FL_WIDTH / 2) - ((FL_WIDTH / 8) / 2)))) / 2;
87         V_CENTERY = (FL_HEIGHT - (FL_HEIGHT / 2.5)) * 0.8;
88         VR_POSX = ((((FL_WIDTH / 8) + (FL_WIDTH / 6)) + (((
      FL_WIDTH / 2) - ((FL_WIDTH / 8) / 2)))) / 2) - (VR_WIDTH / 2);
89         VR_POSY = (FL_HEIGHT - (FL_HEIGHT / 2.5)) * 0.8 - (VR_HEIGHT / 2);
90     }
91
92     else if (m_valveName.compare("V13") == 0)
93     {
94         V_CENTERX = ((((FL_WIDTH / 2) - ((FL_WIDTH / 8) / 2)) + (
      FL_WIDTH / 8)) + ((FL_WIDTH - ((FL_WIDTH / 8) + (
      FL_WIDTH / 6))))) / 2;
95         V_CENTERY = (FL_HEIGHT - (FL_HEIGHT / 2.5)) * 0.6;
96         VR_POSX = (((((FL_WIDTH / 2) - ((FL_WIDTH / 8) / 2)) + (
      FL_WIDTH / 8)) + ((FL_WIDTH - ((FL_WIDTH / 8) + (
      FL_WIDTH / 6))))) / 2) - (VR_WIDTH / 2);
97         VR_POSY = (FL_HEIGHT - (FL_HEIGHT / 2.5)) * 0.6 - (VR_HEIGHT / 2);
98     }
99
100     else if (m_valveName.compare("V23") == 0)
101     {
102         V_CENTERX = ((((FL_WIDTH / 2) - ((FL_WIDTH / 8) / 2)) + (
      FL_WIDTH / 8)) + ((FL_WIDTH - ((FL_WIDTH / 8) + (
      FL_WIDTH / 6))))) / 2;
103         V_CENTERY = (FL_HEIGHT - (FL_HEIGHT / 2.5)) * 0.9;
104         VR_POSX = (((((FL_WIDTH / 2) - ((FL_WIDTH / 8) / 2)) + (
      FL_WIDTH / 8)) + ((FL_WIDTH - ((FL_WIDTH / 8) + (
      FL_WIDTH / 6))))) / 2) - (VR_WIDTH / 2);
105         VR_POSY = (FL_HEIGHT - (FL_HEIGHT / 2.5)) * 0.9 - (VR_HEIGHT / 2);
106     }
107 }
```

Voici le graphe des appelants de cette fonction :



4.14.3.8 isClose()

```
bool Valve::isClose ( )
```

Connaitre l'état de la vanne.

Renvoie

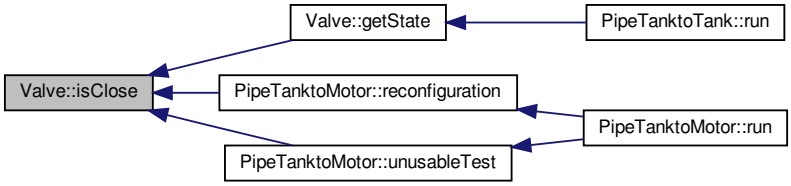
true
false

Définition à la ligne 176 du fichier Valve.cpp.

Référencé par getState(), PipeTanktoMotor : :reconfiguration(), et PipeTanktoMotor : :unusableTest().

```
177 {
178     if (m_stateValve == CLOSE)
179         return true;
180     return false;
181 }
```

Voici le graphe des appelants de cette fonction :



4.14.3.9 open()

```
void Valve::open ( )
```

Ouvre la vanne.

Définition à la ligne 28 du fichier Valve.cpp.

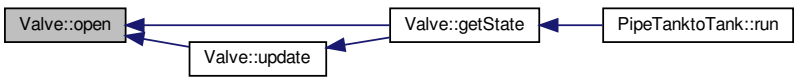
Références initMemberOpenGI().

Référencé par getState(), et update().

```
29 {
30     initMemberOpenGI();
31     m_stateValve = OPEN;
32 }
```

Voici le graphe d'appel pour cette fonction :

Voici le graphe des appelants de cette fonction :



4.14.3.10 push_in_map()

```
void Valve::push_in_map (
            Tank * tank,
            Motor * motor )
```

Réservoir to Vanne to Moteur.

Paramètres

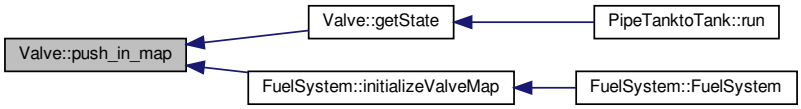
tank	
motor	

Définition à la ligne 189 du fichier Valve.cpp.

Référencé par getState(), et FuelSystem : :initializeValveMap().

```
190 {
191     map_Valve[tank] = motor;
192 }
```

Voici le graphe des appelants de cette fonction :



4.14.3.11 update()

```
void Valve::update ( )
```

Met à jour la Vanne.

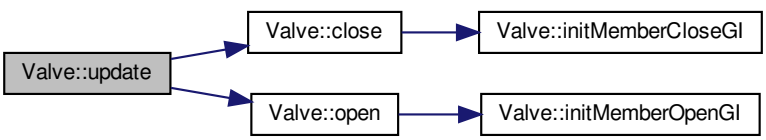
Définition à la ligne 48 du fichier Valve.cpp.

Références close(), et open().

Référencé par getState().

```
49 {
50     if (m_stateValve == OPEN)
51     {
52         close();
53     }
54     else
55         open();
56 }
```

Voici le graphe d'appel pour cette fonction :



Voici le graphe des appelants de cette fonction :

La documentation de cette classe a été générée à partir des fichiers suivants :
- include/Valve.h
- src/Valve.cpp

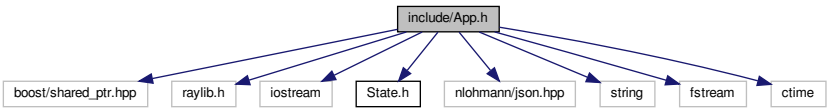
Chapitre 5

Documentation des fichiers

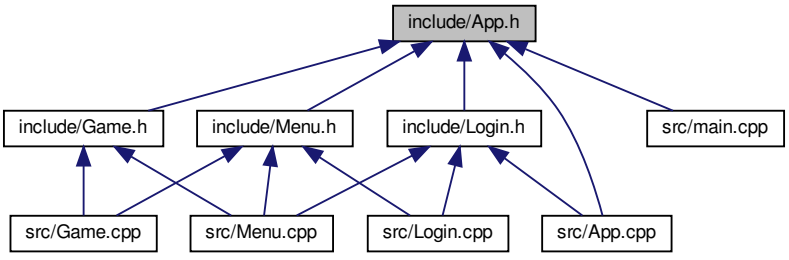
5.1 Référence du fichier include/App.h

```
#include <boost/shared_ptr.hpp>
#include <raylib.h>
#include <iostream>
#include "State.h"
#include <nlohmann/json.hpp>
#include <string>
#include <fstream>
#include <ctime>
```

Graphe des dépendances par inclusion de App.h :



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



Classes

— class App

Définitions de type

— using json = nlohmann : :json

5.1.1 Documentation des définitions de type

5.1.1.1 json

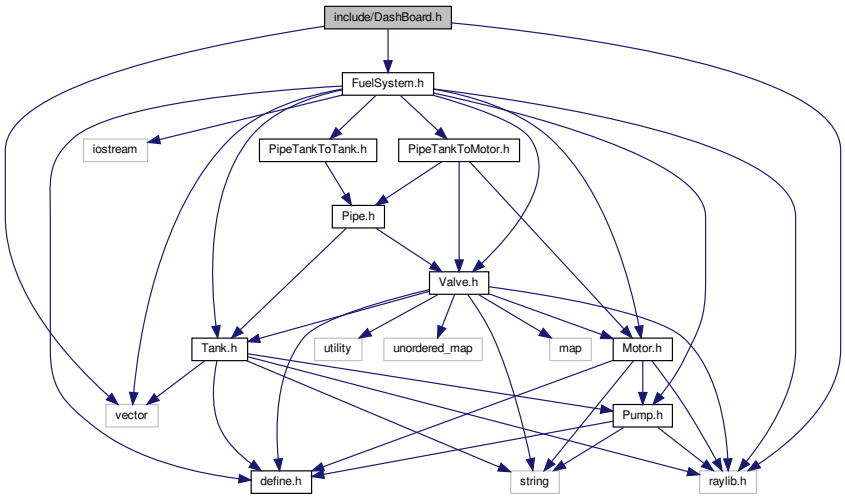
```
using json = nlohmann::json
```

Définition à la ligne 9 du fichier App.h.

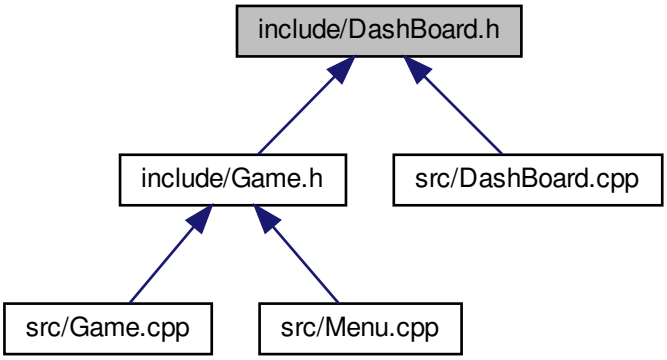
5.2 Référence du fichier include/DashBoard.h

```
#include <vector>
#include <raylib.h>
#include "FuelSystem.h"
```

Graphe des dépendances par inclusion de DashBoard.h :



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :

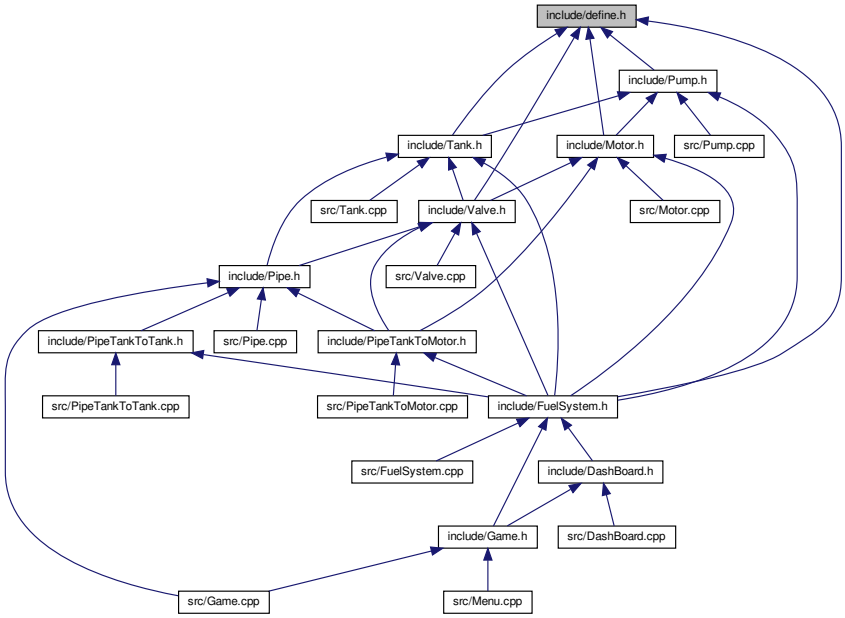


Classes

— class DashBoard

5.3 Référence du fichier include/define.h

Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



Variables

— const int FL_WIDTH = 800
— const int FL_HEIGHT = 700

5.3.1 Documentation des variables

5.3.1.1 FL_HEIGHT

```
const int FL_HEIGHT = 700
```

Définition à la ligne 5 du fichier define.h.

Référencé par DashBoard : :createButton(), FuelSystem : :drawPipe(), Valve : :initMemberCloseGl(), Pump : :init↩MemberGl(), et Valve : :initMemberOpenGl().

5.3.1.2 FL_WIDTH

```
const int FL_WIDTH = 800
```

Définition à la ligne 4 du fichier define.h.

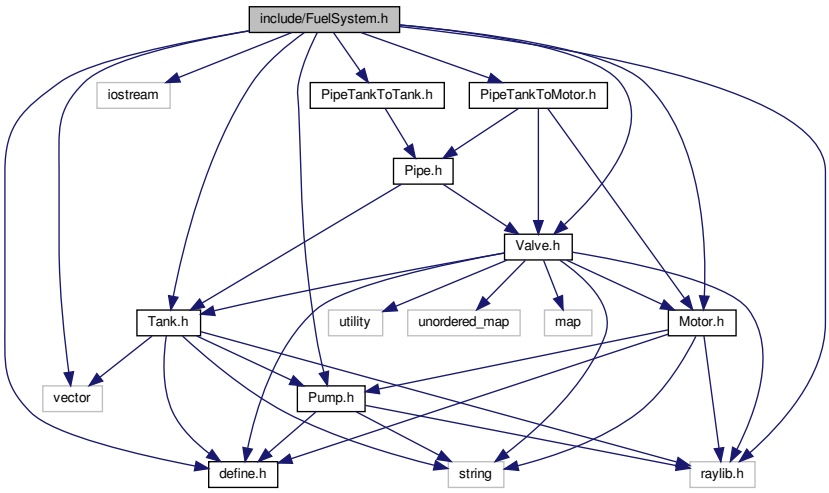
Référencé par DashBoard : :createButton(), DashBoard : :createButtonName(), FuelSystem : :drawPipe(), Valve↩: :initMemberCloseGl(), Motor : :initMemberGl(), Pump : :initMemberGl(), Tank : :initMemberGl(), et Valve : :init↩MemberOpenGl().

5.4 Référence du fichier include/FuelSystem.h

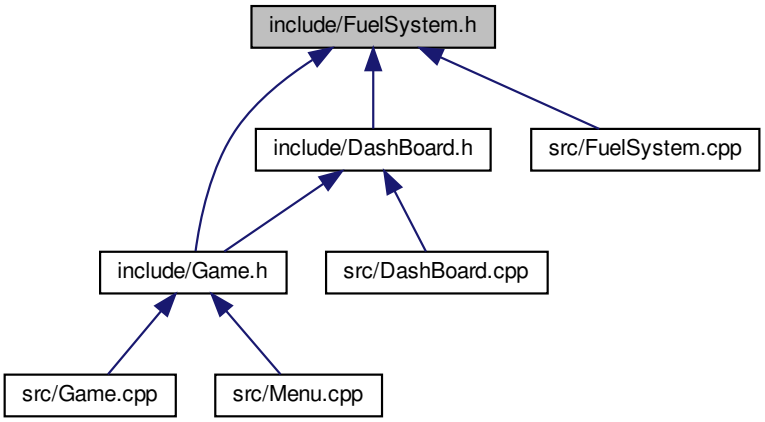
```
#include <raylib.h>
#include <vector>
#include <iostream>
#include "define.h"
#include "Tank.h"
#include "Motor.h"
#include "Valve.h"
#include "Pump.h"
#include "PipeTankToTank.h"
```

```
#include "PipeTankToMotor.h"
```

Graphe des dépendances par inclusion de FuelSystem.h :



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



Classes

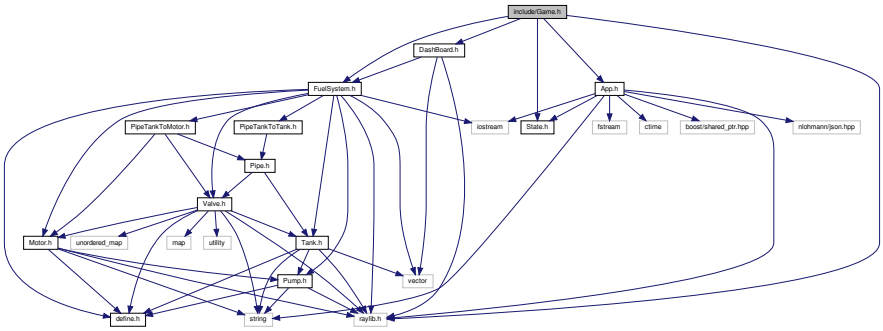
— class FuelSystem

5.5 Référence du fichier include/Game.h

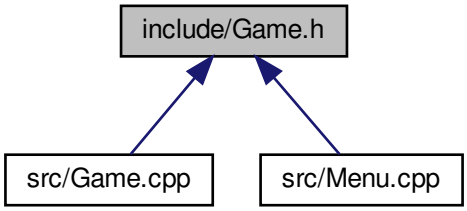
```
#include <raylib.h>
#include "State.h"
```

```
#include "App.h"
#include "FuelSystem.h"
#include "DashBoard.h"
```

Graphe des dépendances par inclusion de Game.h :



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



Classes

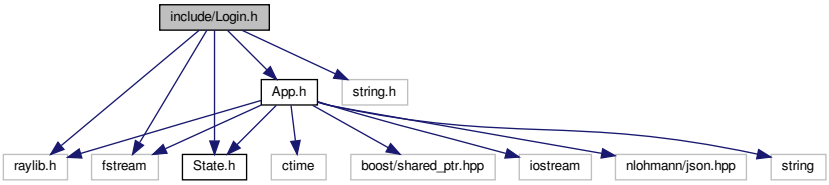
— class Game

5.6 Référence du fichier include/Login.h

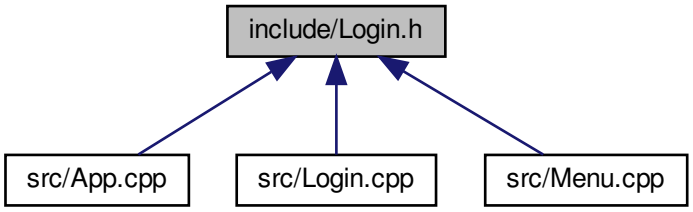
```
#include <raylib.h>
#include <string.h>
#include <fstream>
#include "State.h"
```

```
#include "App.h"
```

Graphe des dépendances par inclusion de Login.h :



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



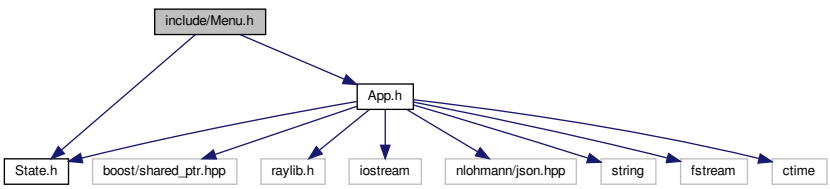
Classes

— class Login

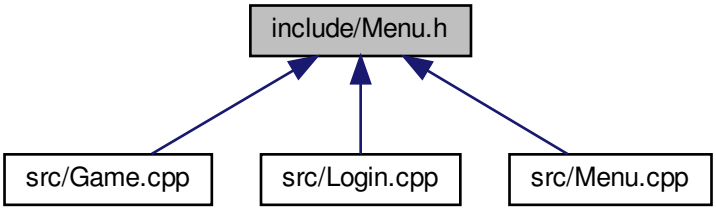
5.7 Référence du fichier include/Menu.h

```
#include "State.h"
#include "App.h"
```

Graphe des dépendances par inclusion de Menu.h :



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



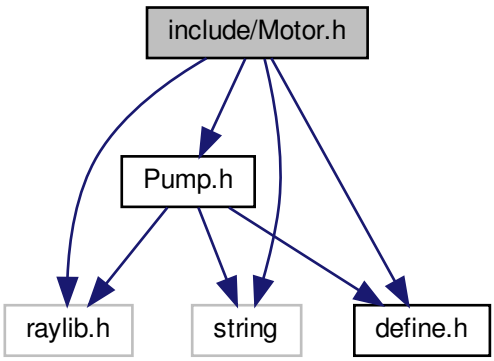
Classes

— class Menu

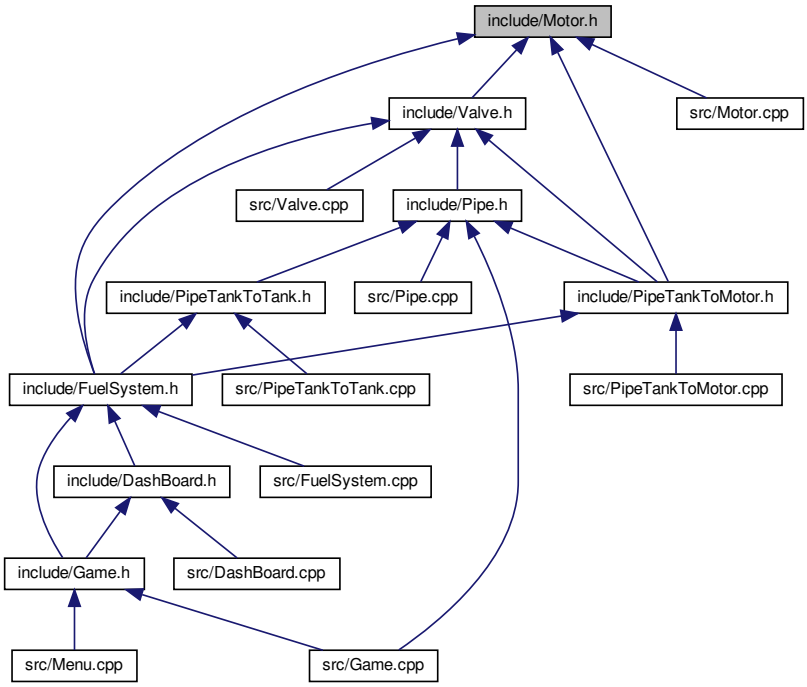
5.8 Référence du fichier include/Motor.h

```
#include <raylib.h>
#include <string>
#include "define.h"
#include "Pump.h"
```

Graphe des dépendances par inclusion de Motor.h :



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



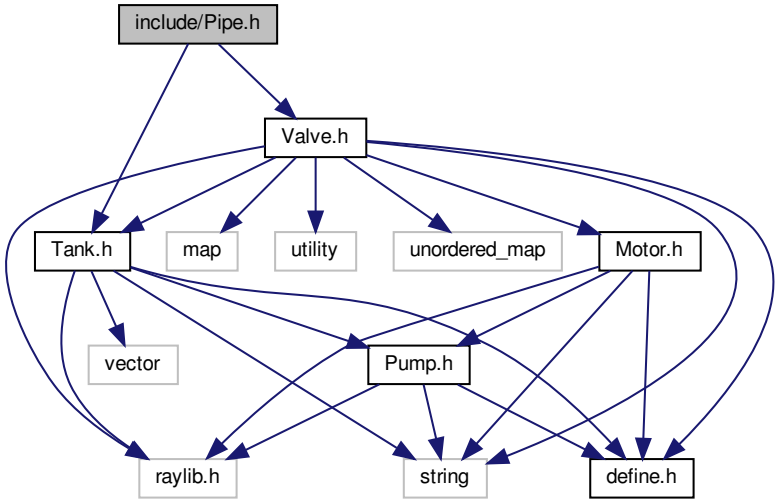
Classes

— class Motor

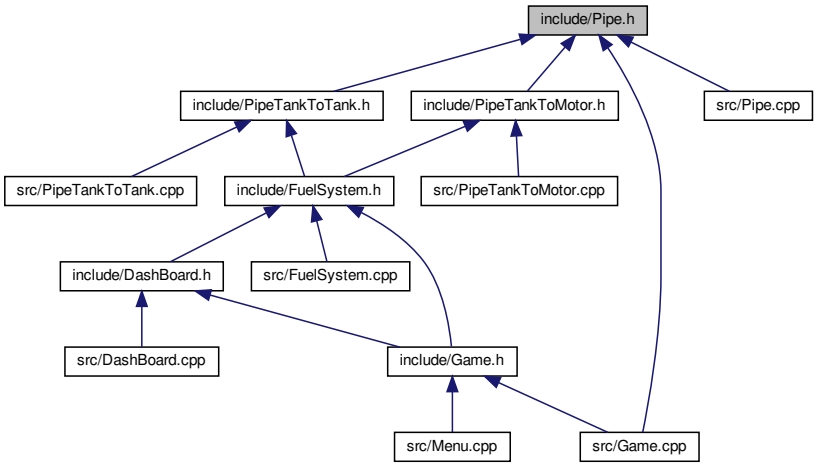
5.9 Référence du fichier include/Pipe.h

```
#include "Tank.h"
#include "Valve.h"
```

Graphe des dépendances par inclusion de Pipe.h :



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



Classes

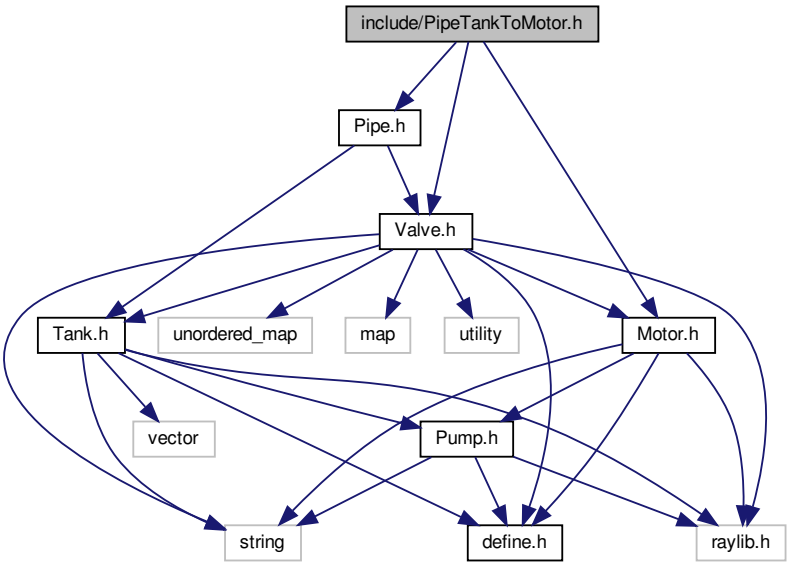
— class Pipe

5.10 Référence du fichier include/PipeTankToMotor.h

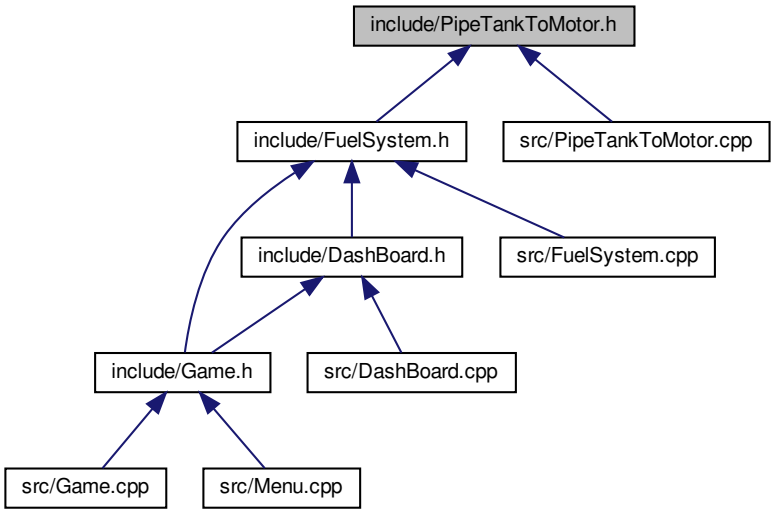
```
#include "Pipe.h"
#include "Valve.h"
```

```
#include "Motor.h"
```

Graphe des dépendances par inclusion de PipeTankToMotor.h :



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



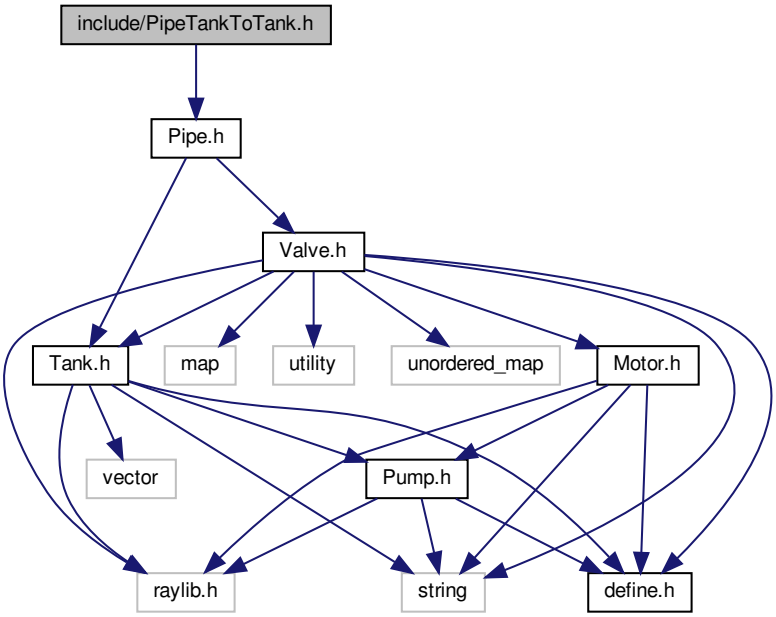
Classes

— class PipeTanktoMotor

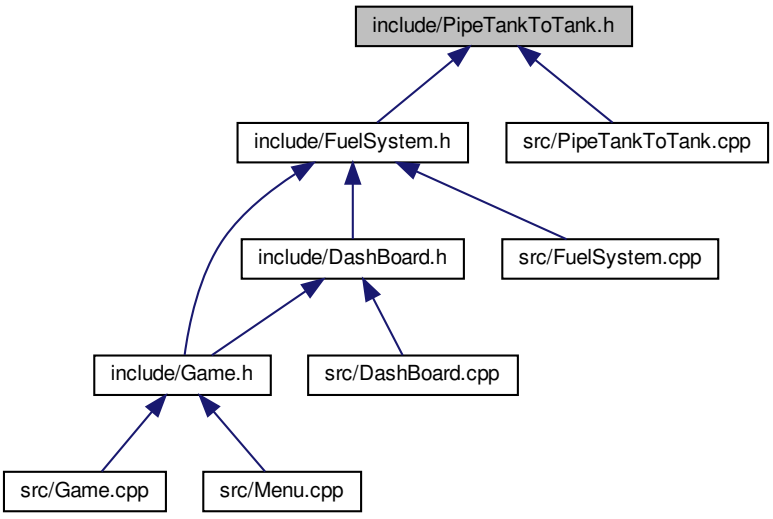
5.11 Référence du fichier include/PipeTankToTank.h

```
#include "Pipe.h"
```

Graphe des dépendances par inclusion de PipeTankToTank.h :



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



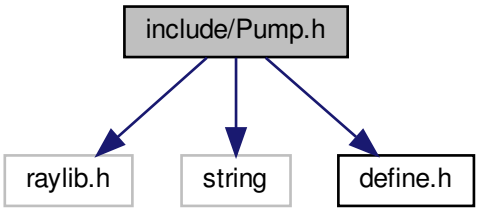
Classes

— class PipeTanktoTank

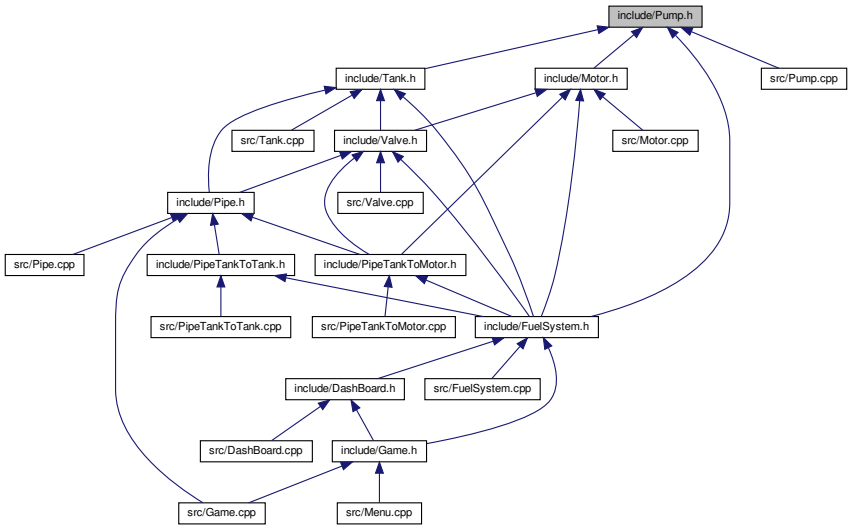
5.12 Référence du fichier include/Pump.h

```
#include <raylib.h>
#include <string>
#include "define.h"
```

Graphe des dépendances par inclusion de Pump.h :



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :

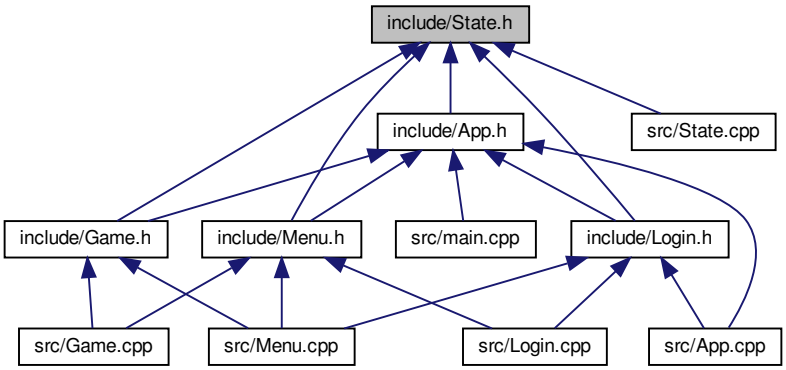


Classes

— class Pump

5.13 Référence du fichier include/State.h

Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



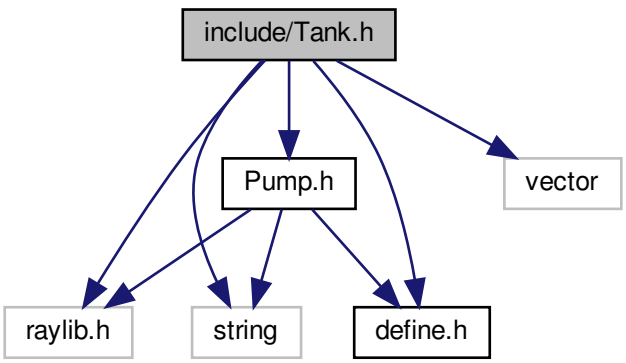
Classes

— class State

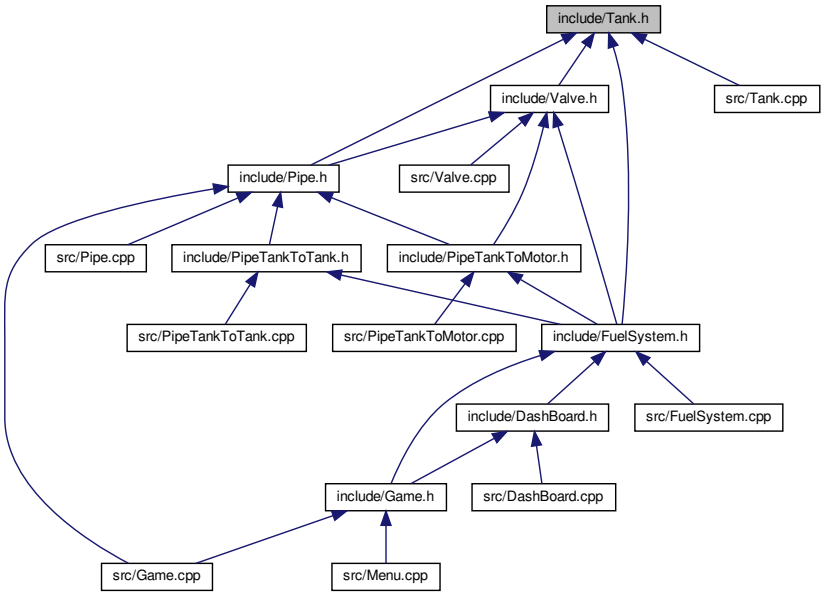
5.14 Référence du fichier include/Tank.h

```
#include <raylib.h>
#include <vector>
#include <string>
#include "define.h"
#include "Pump.h"
```

Graphe des dépendances par inclusion de Tank.h :



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



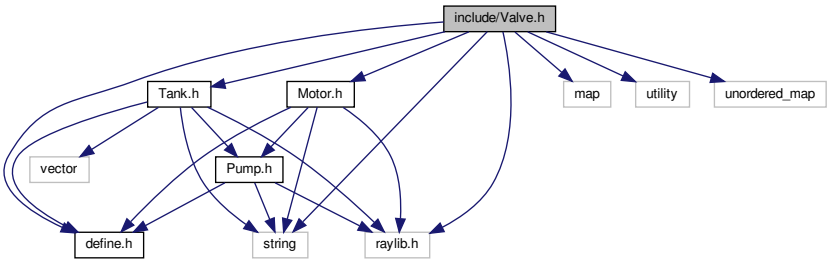
Classes

— class Tank

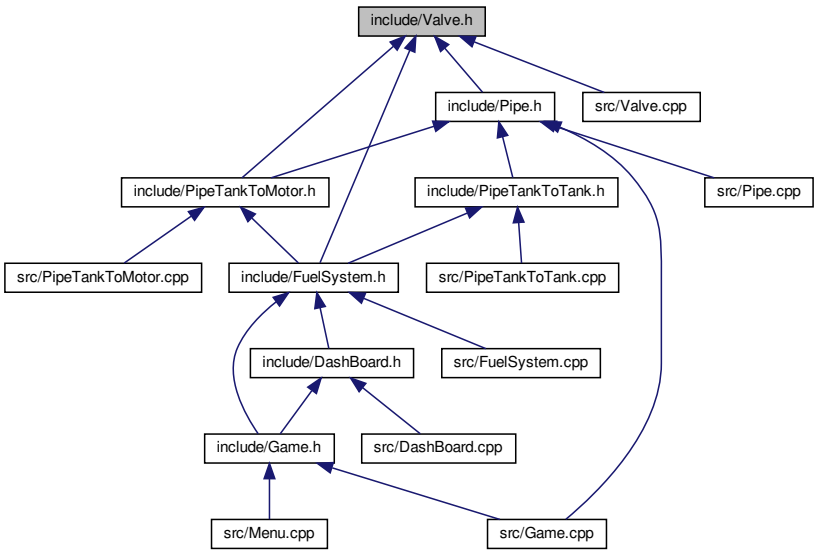
5.15 Référence du fichier include/Valve.h

```
#include "define.h"
#include <raylib.h>
#include <string>
#include <map>
#include <utility>
#include <unordered_map>
#include "Tank.h"
#include "Motor.h"
```

Graphe des dépendances par inclusion de Valve.h :



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



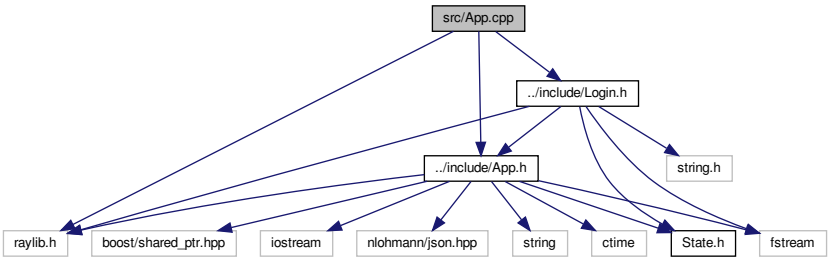
Classes

— class Valve

5.16 Référence du fichier src/App.cpp

```
#include <raylib.h>
#include "../include/App.h"
#include "../include/Login.h"
```

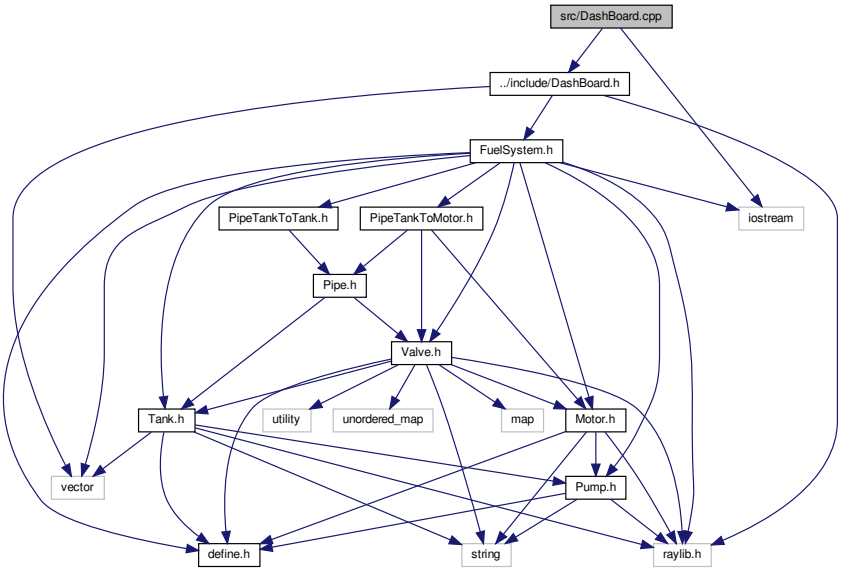
Graphe des dépendances par inclusion de App.cpp :



5.17 Référence du fichier src/DashBoard.cpp

```
#include "../include/DashBoard.h"
#include <iostream>
```

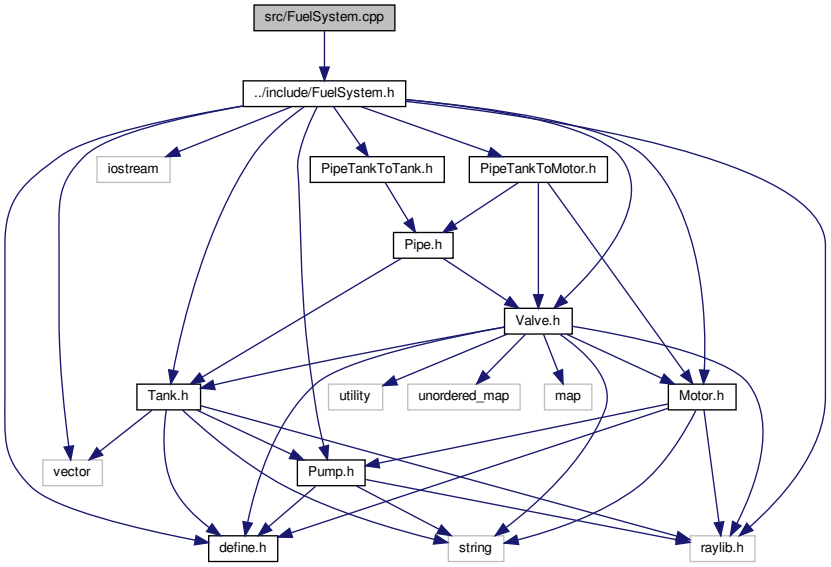
Graphe des dépendances par inclusion de DashBoard.cpp :



5.18 Référence du fichier src/FuelSystem.cpp

```
#include "../include/FuelSystem.h"
```

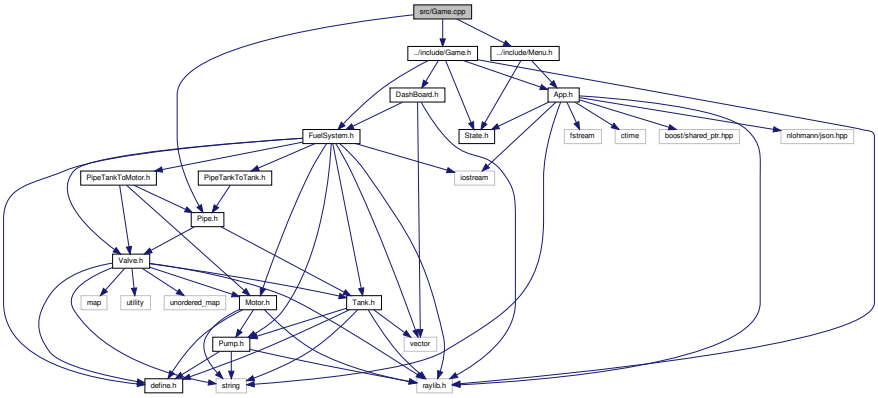
Graphe des dépendances par inclusion de FuelSystem.cpp :



5.19 Référence du fichier src/Game.cpp

```
#include "../include/Game.h"
#include "../include/Pipe.h"
#include "../include/Menu.h"
```

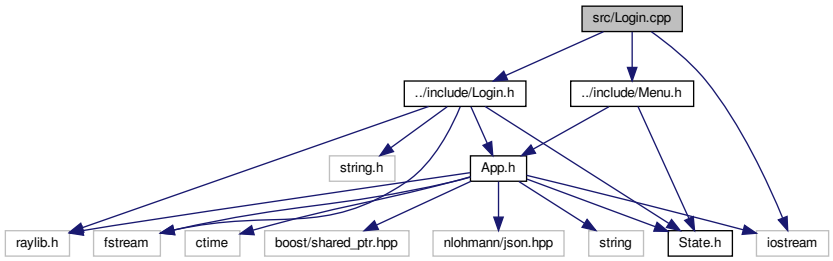
Graphe des dépendances par inclusion de Game.cpp :



5.20 Référence du fichier src/Login.cpp

```
#include "../include/Login.h"
#include "../include/Menu.h"
#include <iostream>
```

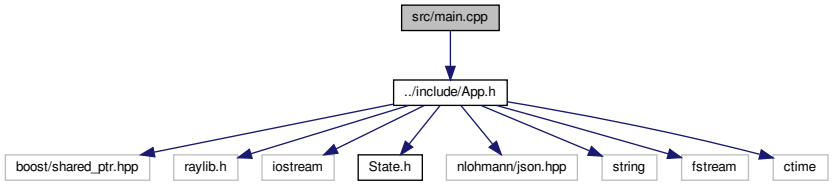
Graphe des dépendances par inclusion de Login.cpp :



5.21 Référence du fichier src/main.cpp

```
#include "../include/App.h"
```

Graphe des dépendances par inclusion de main.cpp :



Fonctions

— int main ()
Main Application.

5.21.1 Documentation des fonctions

5.21.1.1 main()

```
int main ( )
```

Main Application.

Renvoie

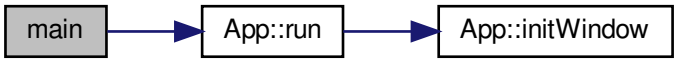
int

Définition à la ligne 8 du fichier main.cpp.

Références App : :run().

```
9 {
10     App app;
11
12     app.run();
13
14     return 0;
15 }
```

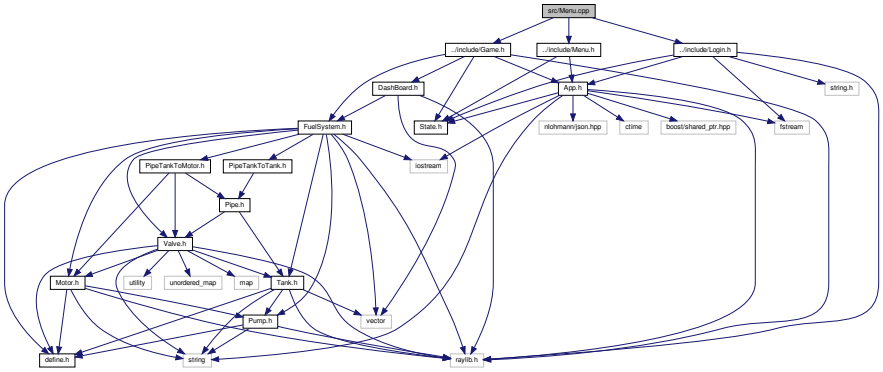
Voici le graphe d'appel pour cette fonction :



5.22 Référence du fichier src/Menu.cpp

```
#include "../include/Menu.h"
#include "../include/Game.h"
#include "../include/Login.h"
```

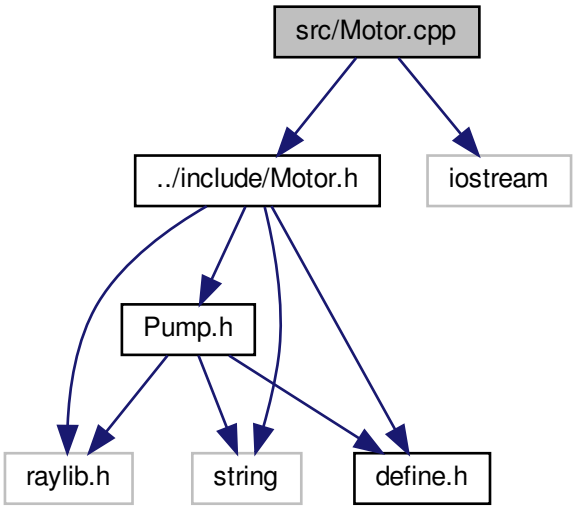
Graphe des dépendances par inclusion de Menu.cpp :



5.23 Référence du fichier src/Motor.cpp

```
#include "../include/Motor.h"
#include <iostream>
```

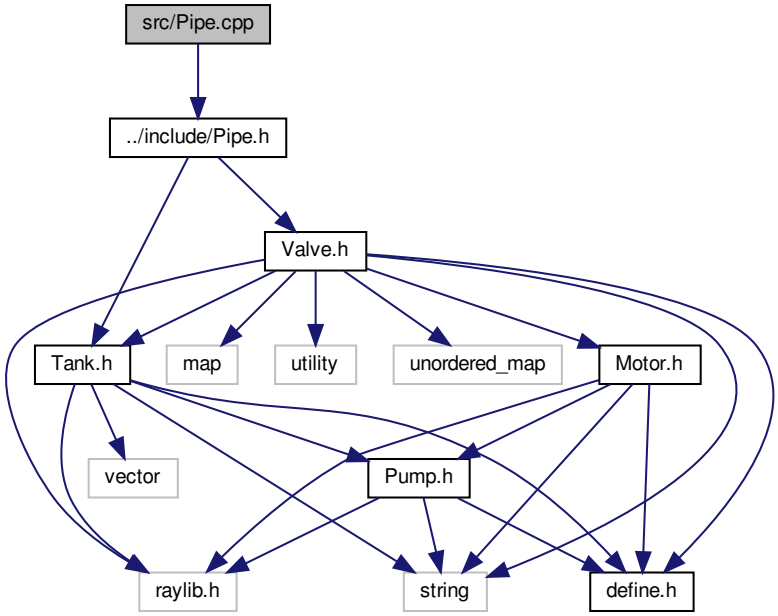
Graphe des dépendances par inclusion de Motor.cpp :



5.24 Référence du fichier src/Pipe.cpp

```
#include "../include/Pipe.h"
```

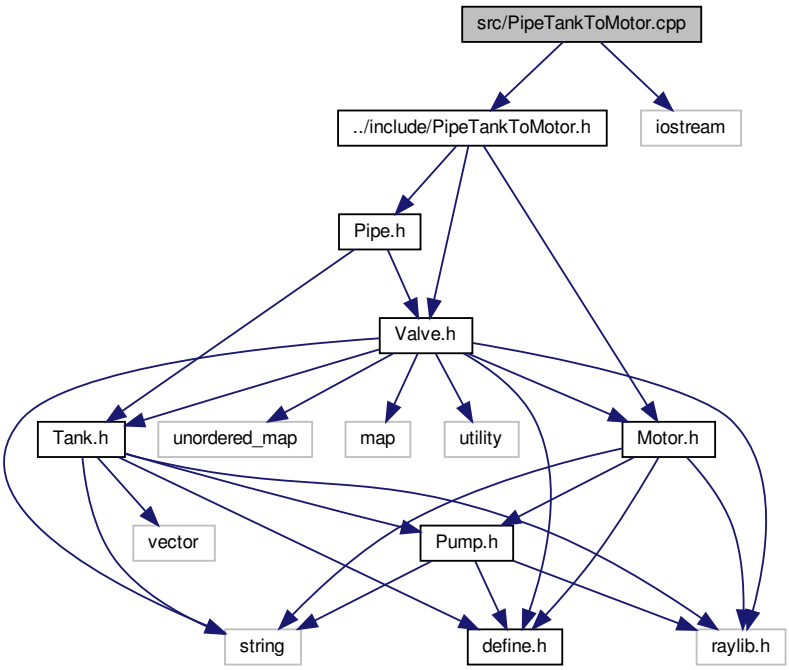
Graphe des dépendances par inclusion de Pipe.cpp :



5.25 Référence du fichier src/PipeTankToMotor.cpp

```
#include "../include/PipeTankToMotor.h"
#include <iostream>
```

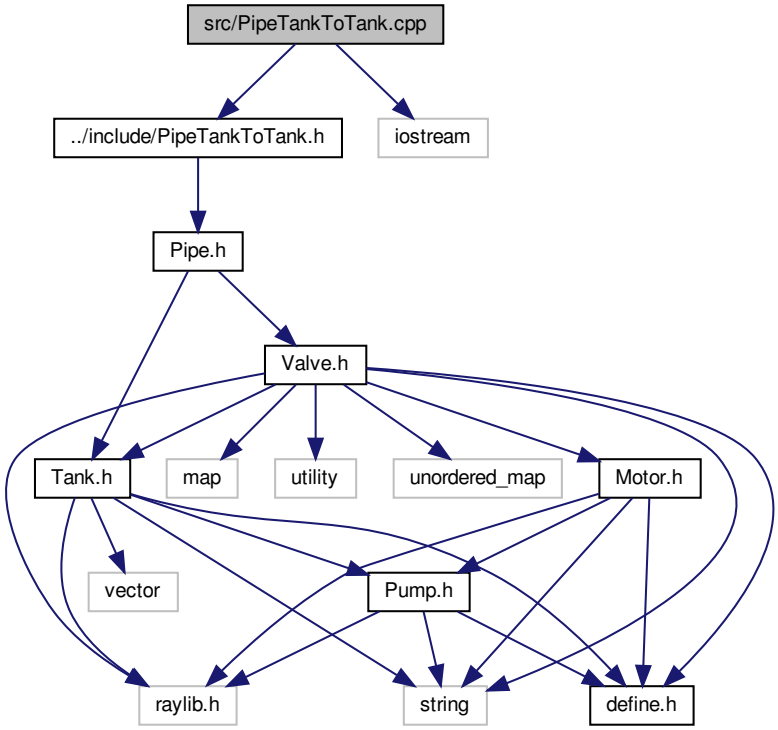
Graphe des dépendances par inclusion de PipeTankToMotor.cpp :



5.26 Référence du fichier src/PipeTankToTank.cpp

```
#include "../include/PipeTankToTank.h"
#include <iostream>
```

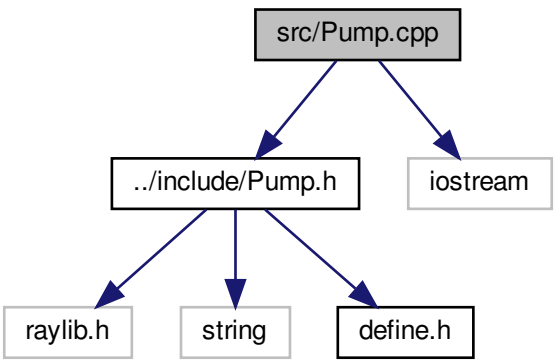
Graphe des dépendances par inclusion de PipeTankToTank.cpp :



5.27 Référence du fichier src/Pump.cpp

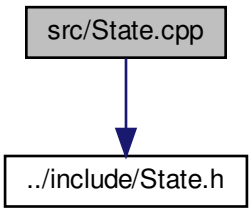
```
#include "../include/Pump.h"
#include <iostream>
```

Graphe des dépendances par inclusion de Pump.cpp :



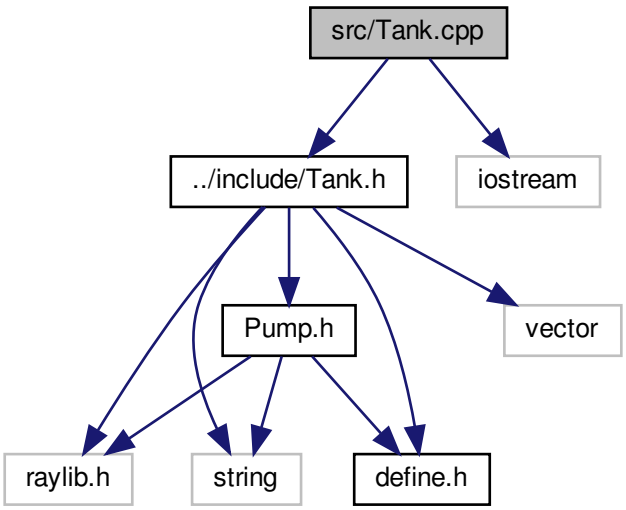
5.28 Référence du fichier src/State.cpp

```
#include "../include/State.h"
```
Graphe des dépendances par inclusion de State.cpp :



5.29 Référence du fichier src/Tank.cpp

```
#include "../include/Tank.h"
#include <iostream>
```
Graphe des dépendances par inclusion de Tank.cpp :



Énumérations

- enum StateTank { FULL, EMPTY, FLOW, STOP }
  Etat des réservoirs.
- enum StatePump { WORKING, STOPPED, UNUSABLE }
  Etat des pompes.

5.29.1 Documentation du type de l'énumération

5.29.1.1 StatePump

```
enum StatePump
```

Etat des pompes.

Valeurs énumérées

WORKING	
STOPPED	
UNUSABLE	

Définition à la ligne 20 du fichier Tank.cpp.

```
21 {
22     WORKING,
23     STOPPED,
24     UNUSABLE
25 };
```

5.29.1.2 StateTank

```
enum StateTank
```

Etat des réservoirs.

Valeurs énumérées

FULL	
EMPTY	
FLOW	
STOP	

Définition à la ligne 8 du fichier Tank.cpp.

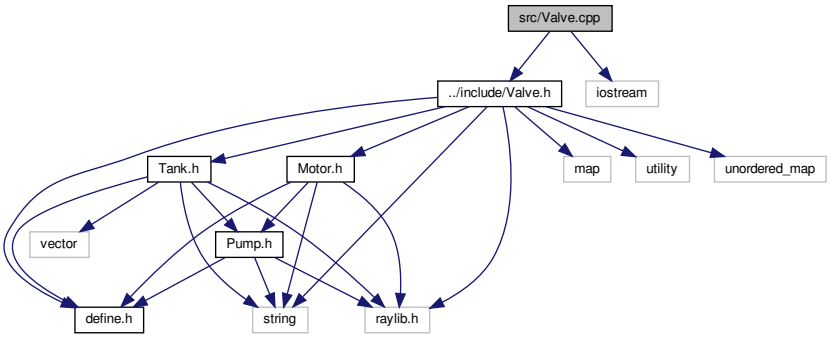
```
9 {
10     FULL,
11     EMPTY,
12     FLOW,
13     STOP
14 };
```

5.30 Référence du fichier src/Valve.cpp

```
#include "../include/Valve.h"
```

```
#include <iostream>
```

Graphe des dépendances par inclusion de Valve.cpp :



Index