My Project

Generated by Doxygen 1.8.8

Thu Jul 14 2016 03:35:32

Contents

Chapter 1

LICENCE

GNU GENERAL PUBLIC LICENSE Version 3, 29 June 2007

Copyright (C) 2007 Free Software Foundation, Inc. http://fsf.org/ Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program–to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

1. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified

Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

1. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

1. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

1. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

```
a) The work must carry prominent notices stating that you modified
it, and giving a relevant date.

b) The work must carry prominent notices stating that it is
released under this License and any conditions added under section
7.  This requirement modifies the requirement in section 4 to
"keep intact all notices".

c) You must license the entire work, as a whole, under this
License to anyone who comes into possession of a copy.  This
License will therefore apply, along with any applicable section 7
additional terms, to the whole of the work, and all its parts,
regardless of how they are packaged.  This License gives no
permission to license the work in any other way, but it does not
invalidate such permission if you have separately received it.

d) If the work has interactive user interfaces, each must display
Appropriate Legal Notices; however, if the Program has interactive
interfaces that do not display Appropriate Legal Notices, your
work need not make them do so.
```

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a

storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

1. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

```
a) Convey the object code in, or embodied in, a physical product
(including a physical distribution medium), accompanied by the
Corresponding Source fixed on a durable physical medium
customarily used for software interchange.

b) Convey the object code in, or embodied in, a physical product
(including a physical distribution medium), accompanied by a
written offer, valid for at least three years and valid for as
long as you offer spare parts or customer support for that product
model, to give anyone who possesses the object code either (1) a
copy of the Corresponding Source for all the software in the
product that is covered by this License, on a durable physical
medium customarily used for software interchange, for a price no
more than your reasonable cost of physically performing this
conveying of source, or (2) access to copy the
Corresponding Source from a network server at no charge.

c) Convey individual copies of the object code with a copy of the
written offer to provide the Corresponding Source.  This
alternative is allowed only occasionally and noncommercially, and
only if you received the object code with such an offer, in accord
with subsection 6b.

d) Convey the object code by offering access from a designated
place (gratis or for a charge), and offer equivalent access to the
Corresponding Source in the same way through the same place at no
further charge.  You need not require recipients to copy the
Corresponding Source along with the object code.  If the place to
copy the object code is a network server, the Corresponding Source
may be on a different server (operated by you or a third party)
that supports equivalent copying facilities, provided you maintain
clear directions next to the object code saying where to find the
Corresponding Source.  Regardless of what server hosts the
Corresponding Source, you remain obligated to ensure that it is
available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provided
you inform other peers where the object code and Corresponding
Source of the work are being offered to the general public at no
charge under subsection 6d.
```

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is

transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

1. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

```
a) Disclaiming warranty or limiting liability differently from the
terms of sections 15 and 16 of this License; or

b) Requiring preservation of specified reasonable legal notices or
author attributions in that material or in the Appropriate Legal
Notices displayed by works containing it; or

c) Prohibiting misrepresentation of the origin of that material, or
requiring that modified versions of such material be marked in
reasonable ways as different from the original version; or

d) Limiting the use for publicity purposes of names of licensors or
authors of the material; or

e) Declining to grant rights under trademark law for use of some
trade names, trademarks, or service marks; or

f) Requiring indemnification of licensors and authors of that
material by anyone who conveys the material (or modified versions of
it) with contractual assumptions of liability to the recipient, for
any liability that these contractual assumptions directly impose on
those licensors and authors.
```

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

1. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

1. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

1. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

1. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent

infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

1. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

1. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

1. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

1. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. E↩ XCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLI↩ ED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE P↩ ROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

1. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRI↩ GHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CO↩ NSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROG↩ RAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

1. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

```
                END OF TERMS AND CONDITIONS

        How to Apply These Terms to Your New Programs
```

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
PolytechWars : Petit simulateur du vaisseau de Yann Solo (Projet de 4e année)
Copyright (C) 2016  Raphaël BRESSON, Mahdi Hammouche

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see <http://www.gnu.org/licenses/>.
```

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

```
PolytechWars Copyright (C) 2016 Raphaël BRESSON, Mahdi Hammouche
This program comes with ABSOLUTELY NO WARRANTY; for details type ‘show w’.
This is free software, and you are welcome to redistribute it
under certain conditions; type ‘show c’ for details.
```

The hypothetical commands ‘show w’ and ‘show c’ should show the appropriate parts of the General Public License. Of course, your program’s commands might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see http://www.gnu.org/licenses/.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read http://www.gnu.org/philosophy/why-not-lgpl.html.

Chapter 2

PolytechWars

Requis

- Linux ou Mac OS (Ne marche pas sur Windows)

Installation

- Installer les bibliothèques à l'aide du script : "install_librairies.sh"

compiler le programme en tapant make

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

Chapter 4

Class Index

4.1 Class List

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

Chapter 5

File Index

5.1 File List

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

Chapter 6

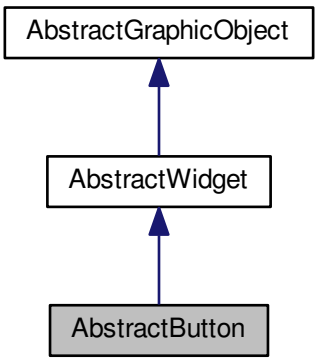
Class Documentation

6.1 AbstractButton Class Reference

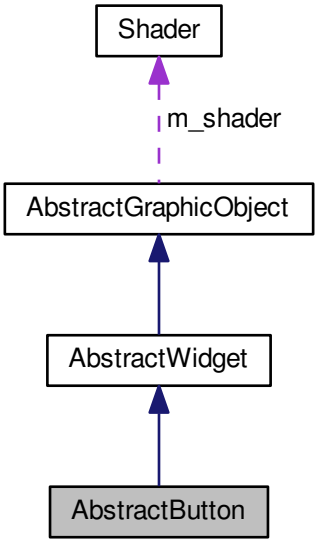
Type polymorphe pour les boutons.

`#include <AbstractButton.h>`

Inheritance diagram for AbstractButton:



Collaboration diagram for AbstractButton:



Public Member Functions

- AbstractButton (Shader *shad, glm::vec2 position, Input *input)

  Constructeur.
- ~AbstractButton ()

  Destructeur.
- virtual void draw (glm::mat4 ortho)=0

  Méthode d'affichage (virtuelle pure)
- void setDimensions (glm::vec2 dim)

  Modificateur de dimensions.
- glm::vec2 getDimensions () const

  Retourne les dimensions.

Protected Member Functions

- void eventHandler ()

  Gère les évènements.
- virtual void onClickEvent ()=0

  Virtuelle pure pour gérer le clic de souris.

Protected Attributes

- bool m_stopThread

  Booléen pour stopper les threads de gestion des évènements.
- std::thread m_threadClickEvent

  Thread de gestion du clic de souris.

- std::condition_variable * m_clicked

  Condition de gestion du clic de souris.
- std::unique_lock< std::mutex > m_clickedLock

  Mutex de gestion du clic de souris.
- glm::vec2 m_dimensions

  Dimensions du bouton.

6.1.1 Detailed Description

Type polymorphe pour les boutons.

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

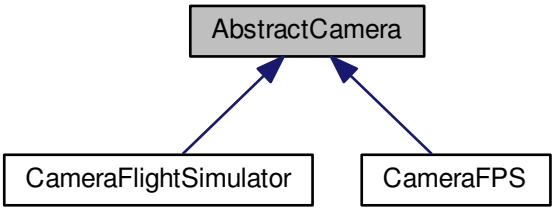
- AbstractButton.h
- AbstractButton.cpp

6.2 AbstractCamera Class Reference

Type polymorphe pour les caméras (point de vue sur le monde 3D)

`#include <AbstractCamera.h>`

Inheritance diagram for AbstractCamera:



Public Member Functions

- AbstractCamera (glm::vec3 const &position, glm::vec3 const &axe_vertical, glm::vec3 const &cible, float proche, float loin, float ratioResolution)

  Construit une caméra et l'initialise en fonction de sa position de son axe vertical et de son point ciblé
- virtual ~AbstractCamera ()

  Destructeur.
- virtual void lookAt ()

  Méthode virtuelle qui doit construire et retourner la matrice de modelview.
- virtual void perspective ()

  Méthode virtuelle pure qui doit construire et retourner la matrice de projection.
- virtual void onEvent (Input const &input)=0

  Méthode virtuelle pure qui traite les évènements pour la caméra appelée à chaque tour de boucle.
- glm::mat4 getProjection () const

  retourne la matrice de projection

- glm::mat4 getModelview () const

  retourne la matrice de modelview
- virtual glm::vec3 getPosition () const =0

  Méthode virtuelle pure qui retourne la position.
- float getVitesse () const

  Retourne la vitesse.
- bool isActive () const

  Savoir si la caméra est active(true) ou non(false)
- void setActive (bool active)

  Déterminer si la caméra est active(true) ou non(false)

Protected Attributes

- glm::vec3 m_position

  Position de la caméra dans le monde 3D.
- glm::vec3 m_axe_vertical

  Axe vertical de la caméra.
- glm::vec3 m_cible

  Point ciblé par la caméra.
- glm::vec3 m_orientation

  Vecteur orientation de la caméra.
- glm::vec3 m_droite

  Vecteur normal à l'orientation et à l'axe vertical -> deplacement lateral.
- float m_proche

  Distance minimale de la caméra pour affichage.
- float m_loin

  Distance maximale de la caméra pour affichage.
- float m_ratioResolution

  Largeur de la fenêtre / Hauteur de la fenêtre.
- float m_vitesse

  Vitesse de déplacement de la caméra.
- glm::mat4 m_projection

  Matrice de projection.
- glm::mat4 m_modelview

  Matrice de modelview.
- bool m_active

  Détermine si la caméra est active ou non.

6.2.1 Detailed Description

Type polymorphe pour les caméras (point de vue sur le monde 3D)

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

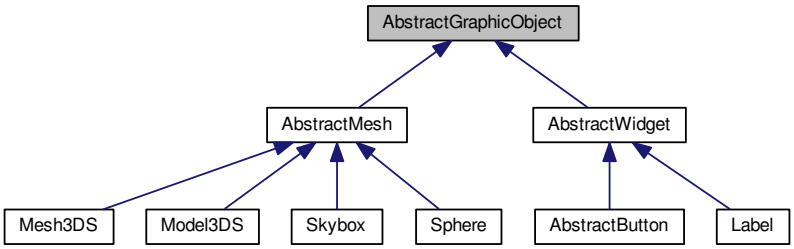
- AbstractCamera.h

6.3 AbstractGraphicObject Class Reference

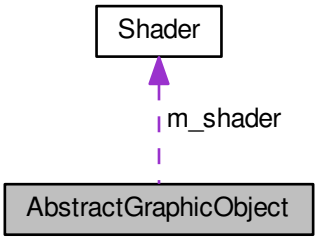
Classe abstraite définissant un type polymorphe pour les objets graphiques.

`#include <AbstractGraphicObject.h>`

Inheritance diagram for AbstractGraphicObject:



Collaboration diagram for AbstractGraphicObject:



Public Member Functions

- AbstractGraphicObject (Shader *shad)

  Construit un objet graphique en utilisant le shader donné en argument pour le rendu.
- virtual ~AbstractGraphicObject ()

  Destructeur.
- virtual void load ()=0

  Génère les objets opengl (vbo,vao,textures) pour cet objet graphique.
- virtual void cleanUp ()

  Détruit les objets de construction intermédiaires.

Protected Attributes

- GLuint m_vboID

  Identifiant OpenGL du Vertex Buffer Object (VBO) (généré lors de l'appel de load())

- GLuint m_vaoID

  Identifiant OpenGL du Vertex Array Object (VAO) (généré lors de l'appel de load())

- Shader * m_shader

  Shader pour le rendu (les shaders sont gérés en externe par les classes GuiFactory -> 2D ou SceneFactory -> 3D)

- std::vector< float > m_vertices

  Tableau temporaire de sommets.

6.3.1 Detailed Description

Classe abstraite définissant un type polymorphe pour les objets graphiques.

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

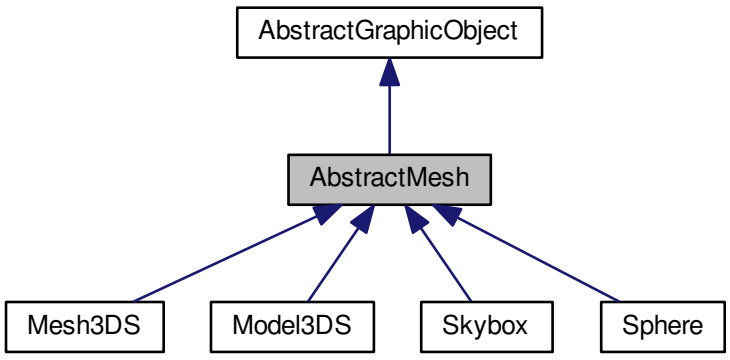
- AbstractGraphicObject.h

6.4 AbstractMesh Class Reference

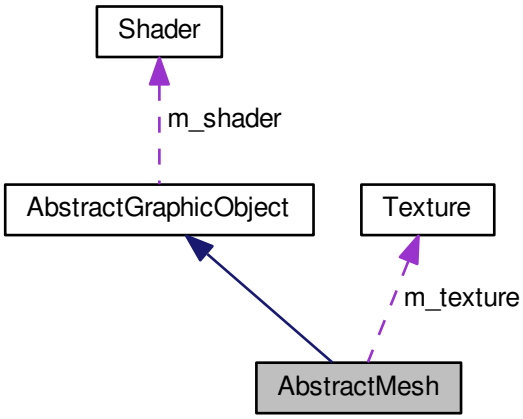
Type polymorphe pour les modèles 3D.

`#include <AbstractMesh.h>`

Inheritance diagram for AbstractMesh:



Collaboration diagram for AbstractMesh:



Public Member Functions

- AbstractMesh (Shader ∗shader, Texture ∗texture)

  Constructeur à partir de la texture donnés en argument.
- AbstractMesh (AbstractMesh ∗mesh)

  Constructeur de copie.
- virtual void load ()

  Construit les objets OpenGL (vbo et vao) pour ce modèle.
- virtual void draw (glm::mat4 mvp)=0

  Méthode virtuelle pure qui affiche le mesh en fonction de la matrice de modelviewProjection.
- virtual void cleanUp ()

  Détruit les objets de construction intermédiaires.

Protected Attributes

- std::vector< float > m_coordTex

  Tableau de coordonnées de textures.
- Texture ∗ m_texture

  Texture de l'objet.
- unsigned long m_nbVertices

  Nombre de vertices de ce mesh.

6.4.1 Detailed Description

Type polymorphe pour les modèles 3D.

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

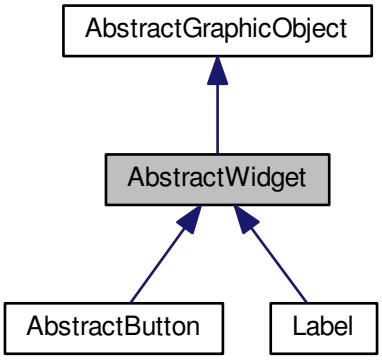
- AbstractMesh.h
- AbstractMesh.cpp

6.5 AbstractWidget Class Reference

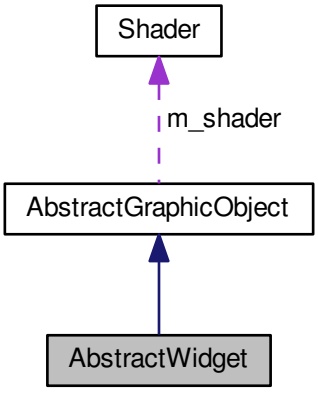
Classe abstraite mère de tous les objets 2D.

`#include <AbstractWidget.h>`

Inheritance diagram for AbstractWidget:



Collaboration diagram for AbstractWidget:



Public Member Functions

- AbstractWidget (Shader *shader, glm::vec2 position)

  Crée un objet 2D à la position sur l'écran indiquée en argument.
- virtual ~AbstractWidget ()

  Détruit un objet2D et libère ses ressources.
- glm::vec2 getPosition ()

Retourne la position d l'objet sur l'écran.

- void setPosition (glm::vec2 const &position)

  Change la position d'un objet 2D sur l'écran.

- virtual void draw (glm::mat4 ortho)=0

  Méthode virtuelle pure d'affichage)

Protected Attributes

- glm::vec2 m_position

  Position de l'objet sur l'écran.

6.5.1 Detailed Description

Classe abstraite mère de tous les objets 2D.

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

- AbstractWidget.h

6.6 ApplicationControl Class Reference

Classe repésentant globalement le programme graphique.

`#include <ApplicationControl.h>`

Public Member Functions

- ApplicationControl (unsigned int windowWidth=800, unsigned int windowHeight=600)

  Crée la fenetre à l'aide des dimensions en argument.

- ~ApplicationControl ()

  Destructeur.

- bool init ()

  initialisation de la SDL, de OpenGL ainsi que de la fenêtre

- bool execute ()

  Exécution du programme graphique.

6.6.1 Detailed Description

Classe repésentant globalement le programme graphique.

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

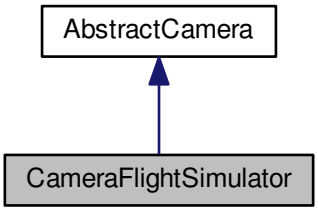
- ApplicationControl.h
- ApplicationControl.cpp

6.7 CameraFlightSimulator Class Reference

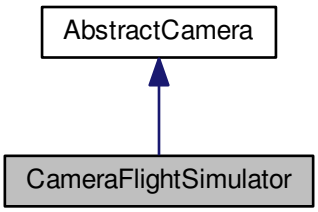
Implémentation de la caméra de simulateur de vol.

`#include <CameraFlightSimulator.h>`

Inheritance diagram for CameraFlightSimulator:



Collaboration diagram for CameraFlightSimulator:



Public Member Functions

- CameraFlightSimulator (glm::vec3 const &position, glm::vec3 const &axe_vertical, glm::vec3 const &cible, float proche, float loin, float ratioResolution, MeshNode *attached)

  Constructeur de caméra de simulateur de vol.
- virtual ~CameraFlightSimulator ()

  Destructeur.
- virtual void onEvent (Input const &input)

  Traitement des évènements clavier.
- virtual glm::vec3 getPosition () const

  Méthode virtuelle qui retourne la position.
- virtual void perspective ()

  Méthode virtuelle pure qui doit construire et retourner la matrice de projection.

Additional Inherited Members

6.7.1 Detailed Description

Implémentation de la caméra de simulateur de vol.

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

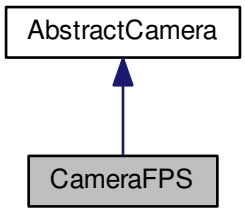
- CameraFlightSimulator.h
- CameraFlightSimulator.cpp

6.8 CameraFPS Class Reference

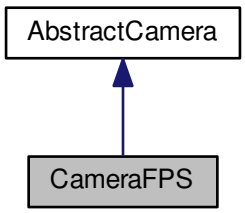
Caméra de type Freefly à deux degrés de liberté en fixant l'axe vertical.

`#include <CameraFPS.h>`

Inheritance diagram for CameraFPS:



Collaboration diagram for CameraFPS:



Public Member Functions

- CameraFPS (glm::vec3 const &position, glm::vec3 const &axe_vertical, glm::vec3 const &cible, float proche, float loin, float ratioResolution, float vitesse, float sensibilite)

  Création et initialisation.
- ~CameraFPS ()

  Destructeur.
- virtual void onEvent (Input const &input)

  Traitement des évènements clavier et souris.
- virtual void perspective ()

  Méthode virtuelle qui doit construire et retourner la matrice de projection pour prendre en compte le zoom.
- virtual glm::vec3 getPosition () const

  retourne la position

Protected Member Functions

- void orienter (int xRel, int yRel)

  Oriente la caméra en fonction du déplacement de la souris.
- void deplacer (glm::vec3 const &deplacement)

  Déplace la caméra.
- void zoomer (int zoom)

  Effectue un zoome dans le monde 3D.

Protected Attributes

- float m_phi

  Angle de rotation par rapport à l'axe vertical.
- float m_theta

  Angle de rotation par rapport au vecteur lateral.
- float m_zoom

  Zoom de la caméra.
- float m_sensibilite

  Vitesse de rotation de la caméra.

6.8.1 Detailed Description

Caméra de type Freefly à deux degrés de liberté en fixant l'axe vertical.

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

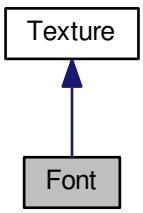
- CameraFPS.h
- CameraFPS.cpp

6.9 Font Class Reference

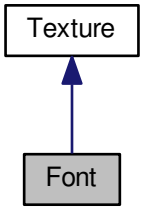
Gère la conversion de texte en texture GL à partir d'une police.

`#include <Font.h>`

Inheritance diagram for Font:



Collaboration diagram for Font:



Public Member Functions

- Font (std::string const &fn_font, unsigned int taille)

  Construit une police à partir du fichier fn_font.
- void createText (std::string const &text, SDL_Color color)

  Crée les objets GL à partir du texte et de la couleur en argument.
- void createTextWithBackground (std::string const &text, SDL_Color textColor, SDL_Color backgroundColor)

  Crée les objets GL à partir du texte et de la couleur en argument avec une couleur de fond.
- ~Font ()

  Destructeur.
- unsigned int getHeight ()

  Retourne la hauteur du texte généré
- unsigned int getWidth ()

  Retourne la largeur du texte généré

Additional Inherited Members

6.9.1 Detailed Description

Gère la conversion de texte en texture GL à partir d'une police.

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

- Font.h
- Font.cpp

6.10 GuiFactory Class Reference

Classe de création d'objets 2D : contient les shaders d'affichage 2D.

`#include <GuiFactory.h>`

Public Member Functions

- GuiFactory ()

  Constructeur.

- ~GuiFactory ()
  Destructeur.
- Label * createLabel (glm::vec2 const &position, std::string const &font)
  Creation d'un label (texte)

6.10.1 Detailed Description

Classe de création d'objets 2D : contient les shaders d'affichage 2D.

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

- GuiFactory.h
- GuiFactory.cpp

6.11 Input Class Reference

gestion des évênements

`#include <Input.h>`

Public Member Functions

- Input (int maxY)
  Constructeur.
- void update ()
  Mise à jour des évênements.
- bool getKey (const SDL_Scancode i) const
  Renvoie true si la touche i est enfoncée false sinon.
- bool getKeyRelease (const SDL_Scancode i) const
  Renvoie true si la touche i est relachée false sinon.
- bool mouseMove () const
  Renvoie true si la souris a bougé false sinon.
- bool getMouseButton (Uint8 i) const
  Renvoie true si le bouton de souris i est enfoncé false sinon.
- bool getMouseButtonRelease (Uint8 i) const
  Renvoie true si le bouton de souris i est relaché false sinon.
- int getXRel () const
  Renvoie le deplacement horizontal de la souris.
- int getYRel () const
  Renvoie le deplacement vertical de la souris.
- int getXAbs () const
  Renvoie la position horizontale de la souris.
- int getYAbs () const
  Renvoie la position verticale de la souris.
- bool terminer () const
  Renvoie true si le programme doit se terminer.
- std::condition_variable * getConditionClick ()
  Retourne un pointeur vers la condition "clic de souris".
- std::mutex * **getClickMutex** ()
- bool getPredicateClick ()
  Retourne la variable de prédicat de clic de souris.

6.11.1 Detailed Description

gestion des évênements

6.11.2 Constructor & Destructor Documentation

6.11.2.1 Input::Input (int *maxY*)

Constructeur.

Destructeur.

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

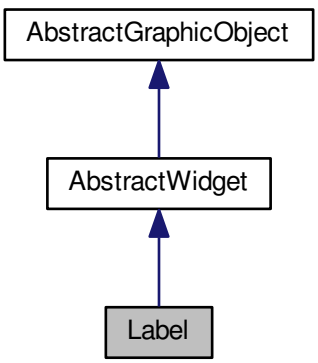
- Input.h
- Input.cpp

6.12 Label Class Reference

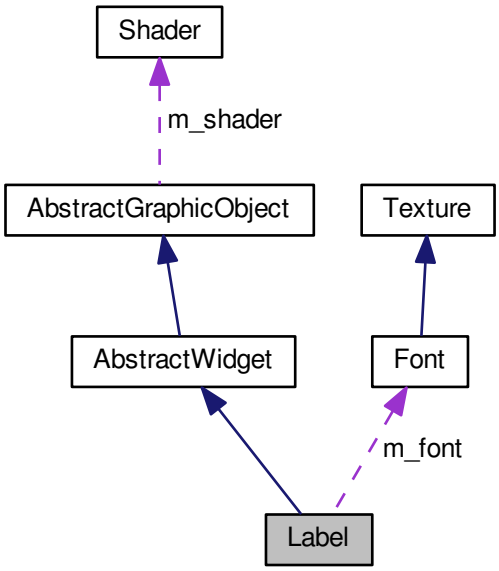
Représente un objet Texte 2D.

`#include <Label.h>`

Inheritance diagram for Label:



Collaboration diagram for Label:



Public Member Functions

- Label (std::string const &font, unsigned int tailleFont, Shader ∗shad, glm::vec2 const &position)

  Constructeur.
- virtual ∼Label ()

  Destructeur.
- void load (GLenum drawingMethod)

  Génération des objets OpenGL.
- virtual void draw (glm::mat4 ortho)

  Affichage du label.
- virtual void cleanUp ()

  Détruit les objets de construction intermédiaires.
- void setText (std::string const &text, SDL_Color const &color, float taille, GLenum drawingMethod)

  Changement du texte et regénération.
- void load ()

  Génère les objets opengl (vbo,vao,textures) pour cet objet graphique.

Protected Attributes

- std::vector< float > m_coordTex

  Tableau temporaire de coordonnées de texture.
- Font m_font

  Gestion du texte.
- unsigned int m_tailleFont

  Taille de la police.

6.12.1 Detailed Description

Représente un objet Texte 2D.

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

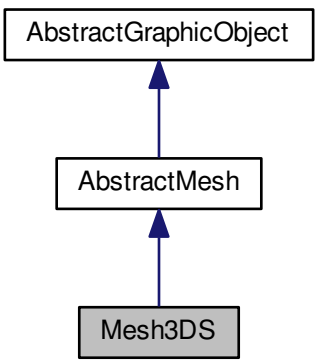
- Label.h
- Label.cpp

6.13 Mesh3DS Class Reference

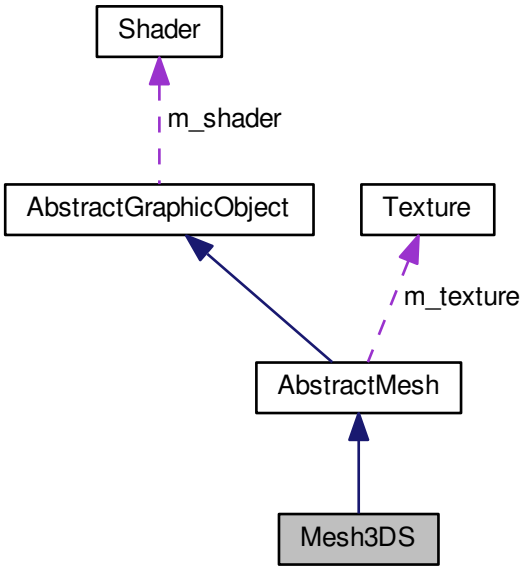
Mesh d'un modèle 3DS.

`#include <Mesh3DS.h>`

Inheritance diagram for Mesh3DS:



Collaboration diagram for Mesh3DS:



Public Member Functions

- Mesh3DS ()

  Constructeur.
- ~Mesh3DS ()

  Destructeur.
- bool extract (Lib3dsFile *file3DS, Lib3dsMesh *mesh)

  Extraction d'un mesh 3DS.
- void extractPoints (Lib3dsMesh *mesh, Lib3dsFace *face)

  Extraction des vertices et coordonnées de texture.
- virtual void load ()

  Chargement des objets OpenGL.
- void draw ()

  Affichage.
- virtual void draw (glm::mat4 mvp)

  Méthode virtuelle pure qui affiche le mesh en fonction de la matrice de modelviewProjection.

Additional Inherited Members

6.13.1 Detailed Description

Mesh d'un modèle 3DS.

6.13.2 Member Function Documentation

6.13.2.1 void Mesh3DS::draw ( )

Affichage.

ne fait rien

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

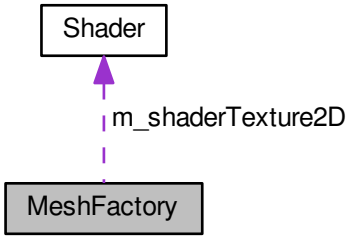
- Mesh3DS.h
- Mesh3DS.cpp

6.14 MeshFactory Class Reference

Classe de gestion des shaders et de génération des modèles 3D.

`#include <MeshFactory.h>`

Collaboration diagram for MeshFactory:



Public Member Functions

- MeshFactory ()

  Constructeur.
- ~MeshFactory ()

  Destructeur.
- Model3DS * create3DSModel (std::string const &modelName)

  Création d'un modèle à partir d'un fichier 3DS MAX.
- Skybox * createSkybox (std::string const &images)

  Création de l'environnement (simulation de l'espace ici)
- Sphere * createPlanet (float radius, std::string const &texture)

  Retourne un sphère texturée qui représente une planète.

Protected Member Functions

- Texture * getTexture (std::string const &name)

  Retourne directement la texture si elle existe déjà sinon la crée puis la retourne.

Protected Attributes

- Shader * m_shaderTexture2D

  Programme shader pour les modèles utilisant des textures 2D.
- std::map< std::string, Texture * > m_textures

  Textures gérées directement par la factory (tous sauf 3DS)

6.14.1 Detailed Description

Classe de gestion des shaders et de génération des modèles 3D.

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

- MeshFactory.h
- MeshFactory.cpp

6.15 MeshNode Class Reference

Classe de gestion de modèles 3D.

`#include <MeshNode.h>`

Public Member Functions

- MeshNode (AbstractMesh *mesh, glm::vec3 position, glm::vec3 orientation=glm::vec3(0, 0, 0))

  Constructeur.
- ~MeshNode ()

  Constructeur.
- void setPosition (glm::vec3 const &position)

  Changement de position.
- void setOrientation (glm::vec3 const &orientation)

  Changement d'orientation.
- void setRotationMatrix (glm::mat4 const &rotMat)

  Changement de la matrice de rotation.
- void draw (glm::mat4 const &projection, glm::mat4 const &modelview)

  Affiche le modèle 3D associé
- glm::vec3 getPosition () const

  Retourne la position du MeshNode.
- glm::vec3 getOrientation () const

  retourne l'orientation du MeshNode

6.15.1 Detailed Description

Classe de gestion de modèles 3D.

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

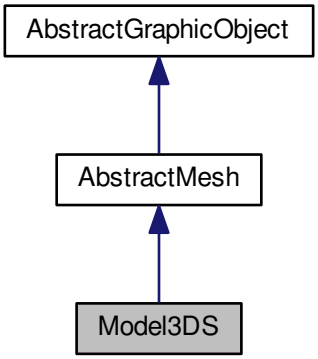
- MeshNode.h
- MeshNode.cpp

6.16 Model3DS Class Reference

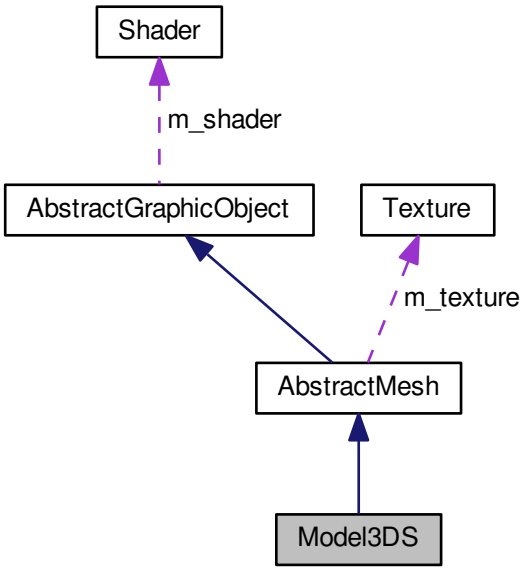
Gestion des modèle 3DS.

`#include <Model3DS.h>`

Inheritance diagram for Model3DS:



Collaboration diagram for Model3DS:



Public Member Functions

- Model3DS (std::string const &file_3ds, Shader *shader)

  Constructeur à partir du fichier 3ds et d'un shader.
- virtual ~Model3DS ()

  Destructeur.
- virtual void load ()

  Construit les objets OpenGL (vbo et vao) pour ce modèle.
- virtual void draw (glm::mat4 mvp)

  Affichage.

Additional Inherited Members

6.16.1 Detailed Description

Gestion des modèle 3DS.

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

- Model3DS.h
- Model3DS.cpp

6.17 SceneManager Class Reference

Gestionnaire des modèle, des caméras (pour le 3D) et de l'affichage.

`#include <SceneManager.h>`

Public Member Functions

- SceneManager (int h)

  Constucteur.
- virtual ~SceneManager ()

  Constucteur.
- MeshNode * addMeshNode (AbstractMesh *mesh, glm::vec3 const &position, glm::vec3 const &orientation=glm::vec3(0, 0, 0))

  Ajout d'un modèle 3D dans la scene, renvoie le MeshNode créé
- void addWidget (AbstractWidget *wid)

  Ajout d'un modèle 2D sur l'écran.
- void init3D (int w, int h)

  Initialisation des modèles 3D et de la caméra.
- void init2D ()

  Initialisation des objets 2D et de la matrice orthogonale.
- void initSounds ()

  Initialisation des musiques et des effets sonores.
- void drawAll ()

  Affichage de tous les modèles.
- void onPreRender ()

  Avant le rendu.
- bool execute (SDL_Window *window, unsigned int w, unsigned int h)

  Exécution de la boucle principale.

- void updateCameras ()

  Déplace les caméras et met à jour les matrices.
- void changeCamera ()

  Change de caméra active.

6.17.1 Detailed Description

Gestionnaire des modèle, des caméras (pour le 3D) et de l'affichage.

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

- SceneManager.h
- SceneManager.cpp

6.18 Shader Class Reference

Classe de gestion de programmes shaders (compilation, édition de liens, contrôle et destruction)

`#include <Shader.h>`

Public Member Functions

- Shader (std::string const &vertexShader, std::string const &fragmentShader)

  Constructeur à partir d'un fichier vertexShader et d'un fichier fragmentShader.
- Shader (Shader const &shader)

  Constructeur de copie d'un shader.
- virtual ~Shader ()

  Destructeur.
- void load ()

  Récupération des sources, compilation et édition de liens.
- std::string getVertexShader () const

  Retourne le nom du fichier vertexShader.
- std::string getFragmentShader () const

  Retourne le nom du fichier fragmentShader.
- void envoyerMat4 (std::string const &nom, glm::mat4 const &matrice)

  Envoyer une matrice 4x4 comme variable Uniform (commune à toutes les instances de ce shader)
- void begin ()

  Début du rendu : à appeler avant le rendu d'un objet pour utiliser ce shader pour son affichage.
- void end ()

  À appeler après le rendu d'un objet ne plus utiliser ce shader.
- bool isProgram ()

  Renvoie true si le programme shader est valide false sinon.
- GLuint getID ()

  Renvoie l'identifiant OpenGL de ce shader.
- void **loadSimple** ()

6.18.1 Detailed Description

Classe de gestion de programmes shaders (compilation, édition de liens, contrôle et destruction)

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

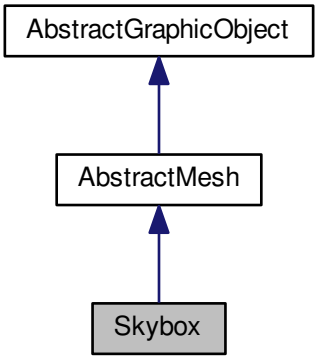
- Shader.h
- Shader.cpp

6.19 Skybox Class Reference

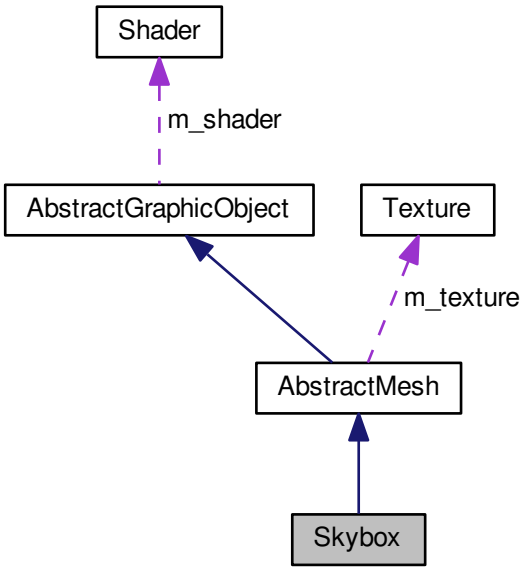
Cube avec textures de ciel plaquées.

`#include <Skybox.h>`

Inheritance diagram for Skybox:



Collaboration diagram for Skybox:



Public Member Functions

- Skybox (float taille, Shader *shad, Texture *tex)

  Constructeur pour une skybox mono-texturée.
- virtual ~Skybox ()

  Destructeur.
- virtual void draw (glm::mat4 mvp)

  Affichage.

Additional Inherited Members

6.19.1 Detailed Description

Cube avec textures de ciel plaquées.

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

- Skybox.h
- Skybox.cpp

6.20 SoundManager Class Reference

Gestionnaire de sons.

`#include <SoundManager.h>`

Public Member Functions

- SoundManager ()

  Constructeur.
- ~SoundManager ()

  Constructeur.
- void addMusic (std::string const &file)

  Ajoute une musique à la liste de lecture.
- void addEffect (std::string const &name, std::string const &file)

  Ajoute un effet à la liste des effets.
- bool isPlayingMusic ()

  Renvoie true si le gectionnaire de son est en train de jouer une musique false sinon.
- void playMusic ()

  Lit la liste de lecture.
- void nextMusic ()

  Lit la musique suivant dans la liste de lecture.
- void playEffect (std::string const &name, int volume)

  Joue l'effet identifié par son nom.

6.20.1 Detailed Description

Gestionnaire de sons.

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

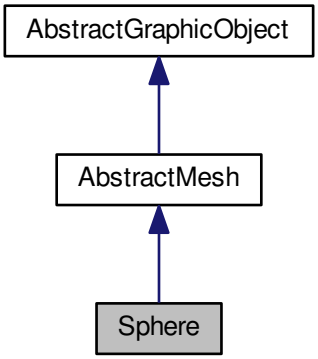
- SoundManager.h
- SoundManager.cpp

6.21 Sphere Class Reference

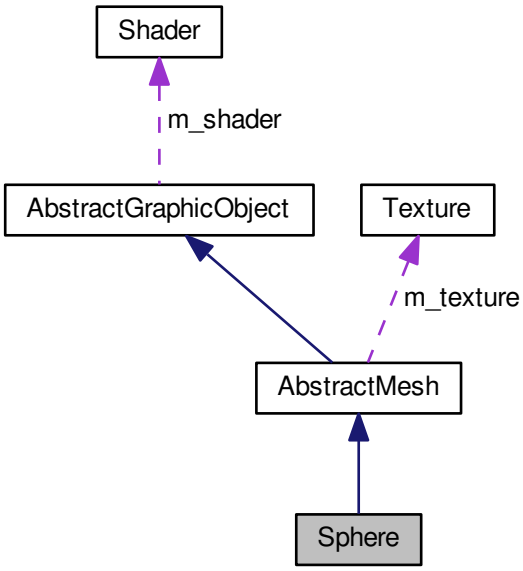
Définition d'une sphère texturée.

`#include <Sphere.h>`

Inheritance diagram for Sphere:



Collaboration diagram for Sphere:



Public Member Functions

- Sphere (float radius, unsigned int nbLat, unsigned int nbLong, Shader *shad, Texture *tex)

  Constructeur.
- ~Sphere ()

  Destructeur.
- void load ()

  Charger les objets GL.
- void initSphere (double r, unsigned int lats, unsigned int longs)

  Initialisation des vertices et des coordonnées de texture.
- void draw (glm::mat4 mvp)

  Affichage.

Additional Inherited Members

6.21.1 Detailed Description

Définition d'une sphère texturée.

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

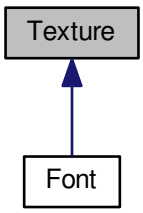
- Sphere.h
- Sphere.cpp

6.22 Texture Class Reference

Gère les textures OpenGL et leur importation via une image ou une police.

`#include <Texture.h>`

Inheritance diagram for Texture:



Public Member Functions

- Texture ()

  Contruit une texture vide.
- Texture (std::string const &fn)

  Construit une texture à partir d'une image.
- Texture (Texture &tex)

Constructeur de copie.

- ~Texture ()

  Destructeur.
- void load ()

  Création des objets GL pour une texture image.
- void loadCullFace ()

  Création des objets GL pour une texture image sans répétition.
- void loadCubemap (std::vector< std::string > faces)

  Création des objets GL pour une texture Cube Map.
- void loadEmpty (unsigned int width, unsigned int height, GLenum format, GLenum formatInterne)

  Création des objets GL pour une texture vide.
- Texture & operator= (Texture &tex)

  Equivalent au constructeur de copie.
- void bind ()

  Verrouillage de la texture (obligatoire pour pouvoir afficher ou modifier les objets GL)
- void unbind ()

  Verrouillage de la texture (Facultatif)
- std::string getName ()

  Renvoie le nom du fichier image ou police.
- GLuint getID ()

  Renvoie l'identifiant OpenGL de la texture.
- bool isValid ()

  Renvoie true si la police est valide false sinon.

Protected Member Functions

- SDL_Surface * readPixels ()

  Retourne une SDL_Surface à parir de l'image désignée par m_fn.
- SDL_Surface * **readPixels** (std::string image)
- SDL_Surface * inversePixels (SDL_Surface *imageSource)

  Inverse les pixels pour retourner une image.
- void uploadToGPU (SDL_Surface *image)

  Charge les objets OpenGL pour cette texture.
- void detectFormat (SDL_Surface *image, GLenum &formatInterne, GLenum &format)

  Détecte le format et le format interne d'une image.

Protected Attributes

- GLuint m_id

  Identifiant OpenGL pour cette texture.
- std::string m_fn

  Nom du fichier image de cette texture.
- GLenum m_format

  Format de l'image.

6.22.1 Detailed Description

Gère les textures OpenGL et leur importation via une image ou une police.

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

- Texture.h
- Texture.cpp

Chapter 7

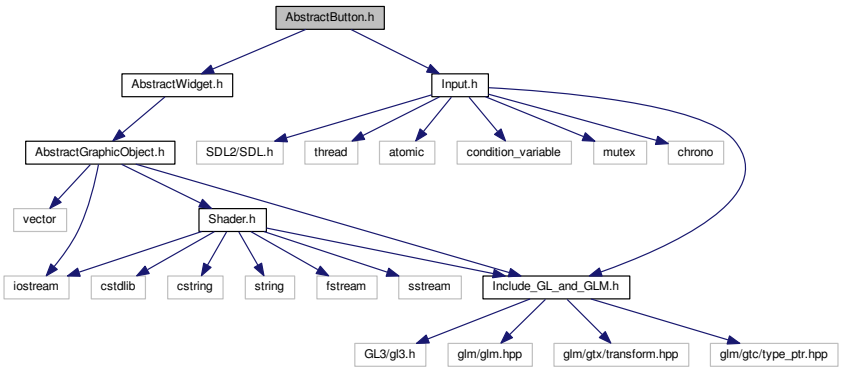
File Documentation

7.1 AbstractButton.h File Reference

Définit un type polymorphe pour les boutons.

```
#include "AbstractWidget.h"
#include "Input.h"
```

Include dependency graph for AbstractButton.h:



Classes

- class AbstractButton

  Type polymorphe pour les boutons.

7.1.1 Detailed Description

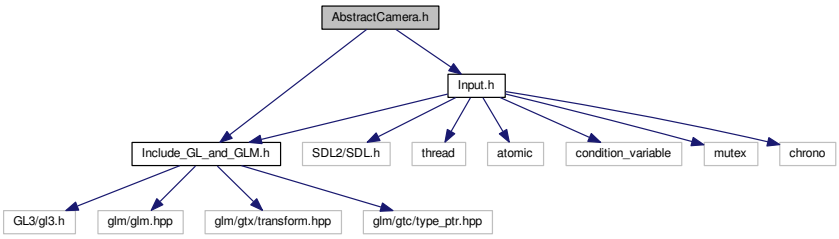
Définit un type polymorphe pour les boutons.

7.2 AbstractCamera.h File Reference

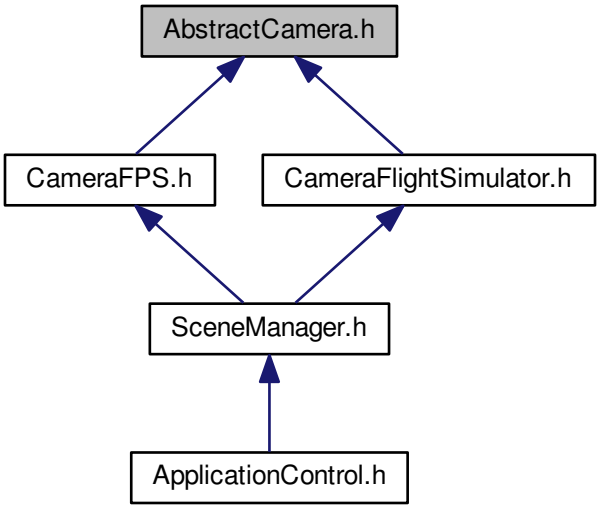
Définit un type polymorphe pour les caméras.

```
#include "Include_GL_and_GLM.h"
#include "Input.h"
```

Include dependency graph for AbstractCamera.h:



This graph shows which files directly or indirectly include this file:



Classes

- class AbstractCamera

  Type polymorphe pour les caméras (point de vue sur le monde 3D)

7.2.1 Detailed Description

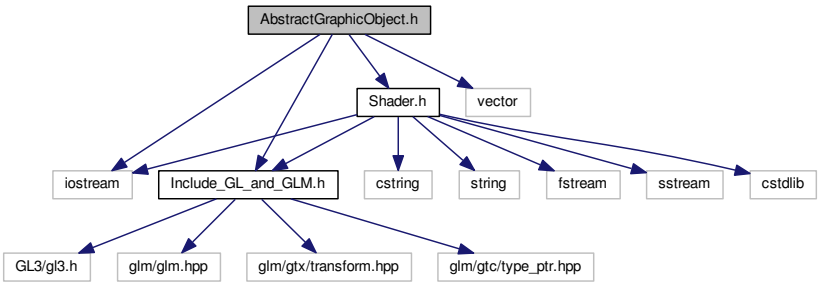
Définit un type polymorphe pour les caméras.

7.3 AbstractGraphicObject.h File Reference

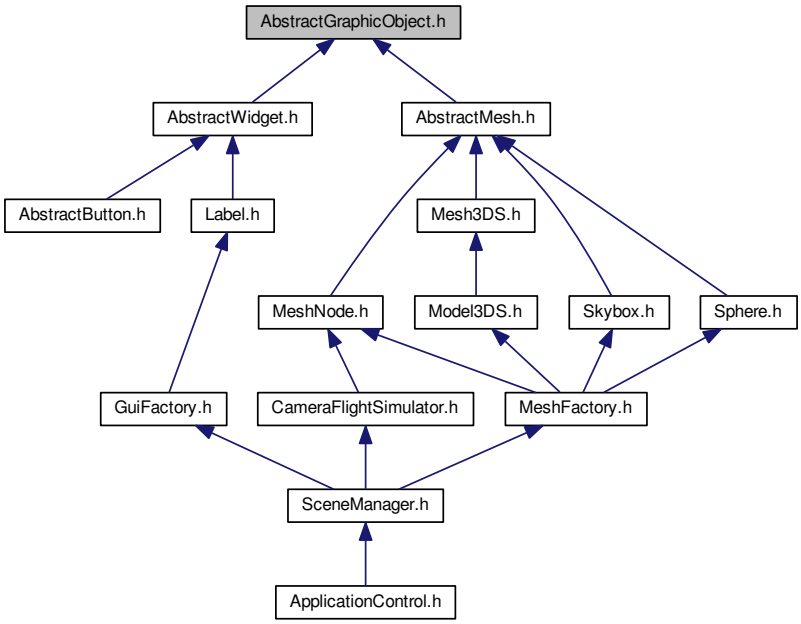
Définit un type polymorphe pour les objets graphiques.

```
#include <iostream>
#include <vector>
#include "Include_GL_and_GLM.h"
#include "Shader.h"
```

Include dependency graph for AbstractGraphicObject.h:



This graph shows which files directly or indirectly include this file:



Classes

- class AbstractGraphicObject

  Classe abstraite définissant un type polymorphe pour les objets graphiques.

Macros

- #define BUFFER_OFFSET(offset) ((void∗)(offset))

Macro de conversion d'un entier non signé en adresse (utile lors de l'initialisation des vao)

7.3.1 Detailed Description

Définit un type polymorphe pour les objets graphiques.

Author

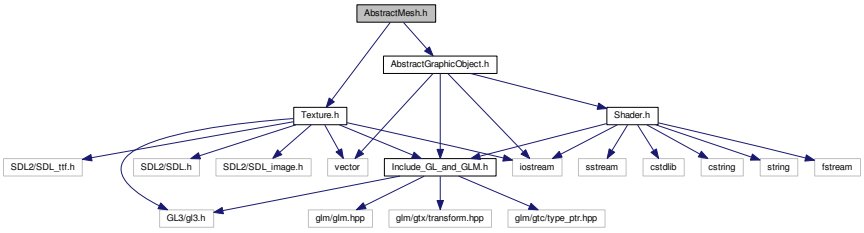
Raphaël BRESSON, Mahdi HAMMOUCHE

7.4 AbstractMesh.h File Reference

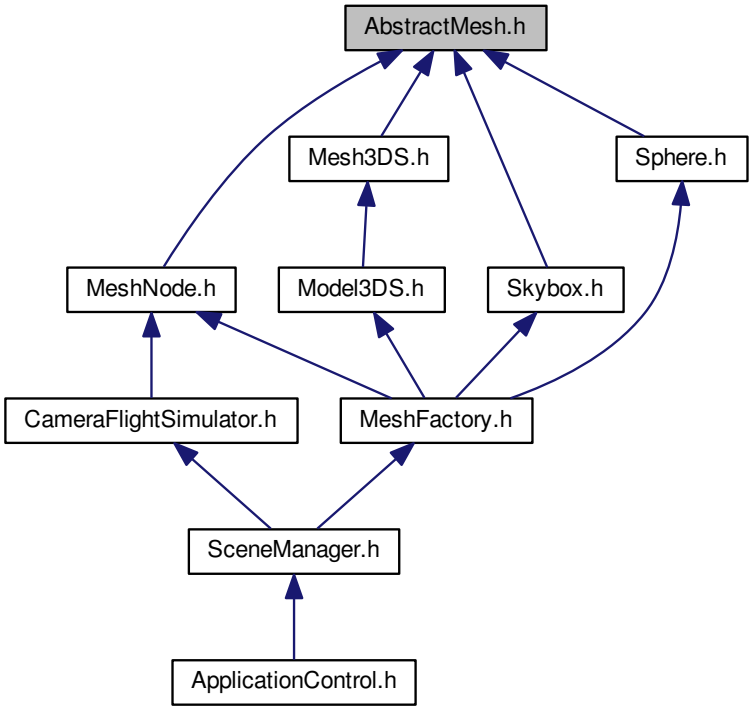
Définition d'un type polymorphe pour les modèles 3D.

```
#include "AbstractGraphicObject.h"
#include "Texture.h"
```

Include dependency graph for AbstractMesh.h:



This graph shows which files directly or indirectly include this file:



Classes

- class AbstractMesh

  Type polymorphe pour les modèles 3D.

7.4.1 Detailed Description

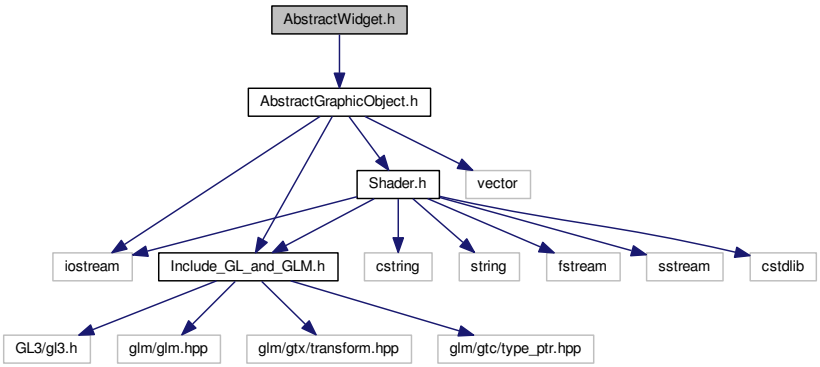
Définition d'un type polymorphe pour les modèles 3D.

7.5 AbstractWidget.h File Reference

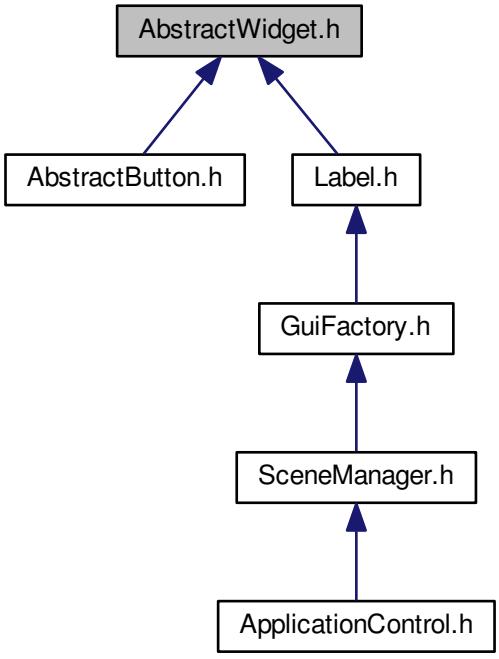
Définit un type polymorphe pour les objets 2D.

```
#include "AbstractGraphicObject.h"
```

Include dependency graph for AbstractWidget.h:



This graph shows which files directly or indirectly include this file:



Classes

- class AbstractWidget

  Classe abstraite mère de tous les objets 2D.

7.5.1 Detailed Description

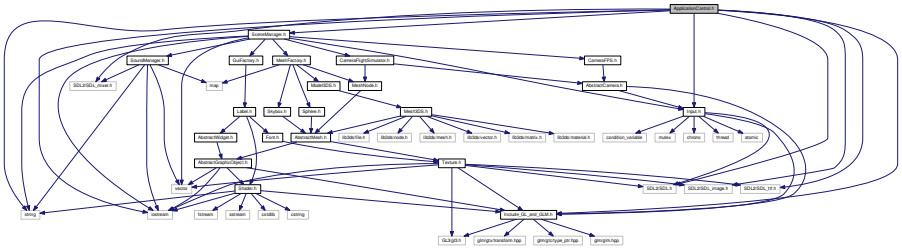
Définit un type polymorphe pour les objets 2D.

7.6 ApplicationControl.h File Reference

Gestion de la SDL et de OpenGL ainsi que du SceneManager.

```
#include <iostream>
#include <string>
#include <SDL2/SDL.h>
#include <SDL2/SDL_image.h>
#include <SDL2/SDL_ttf.h>
#include <SDL2/SDL_mixer.h>
#include "Include_GL_and_GLM.h"
#include "Input.h"
#include "SceneManager.h"
```

Include dependency graph for ApplicationControl.h:



Classes

- class ApplicationControl

  Classe repésentant globalement le programme graphique.

7.6.1 Detailed Description

Gestion de la SDL et de OpenGL ainsi que du SceneManager.

Author

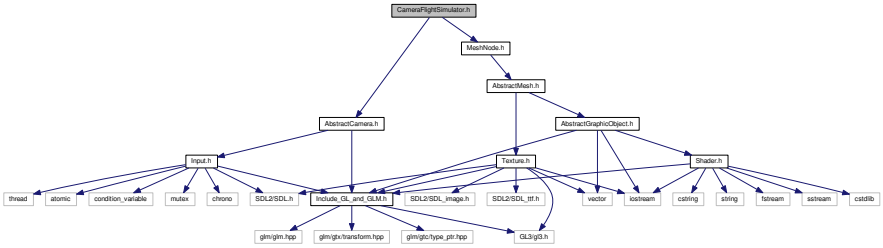
Raphaël BRESSON, Mahdi HAMMOUCHE

7.7 CameraFlightSimulator.h File Reference

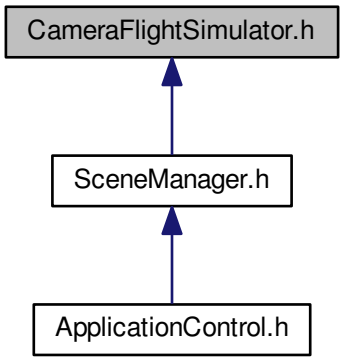
Implémentation de la caméra de simulateur de vol.

```
#include "AbstractCamera.h"
#include "MeshNode.h"
```

Include dependency graph for CameraFlightSimulator.h:



This graph shows which files directly or indirectly include this file:



Classes

- class CameraFlightSimulator

  Implémentation de la caméra de simulateur de vol.

7.7.1 Detailed Description

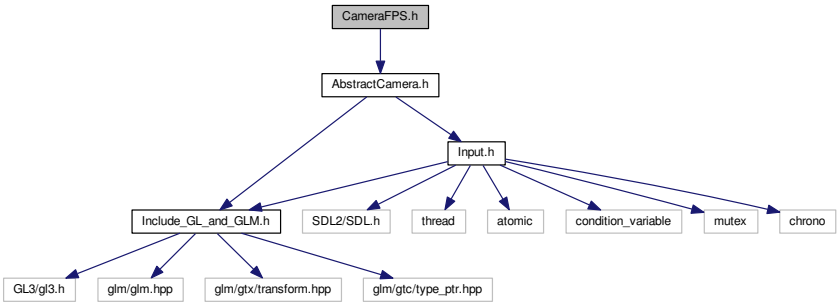
Implémentation de la caméra de simulateur de vol.

7.8 CameraFPS.h File Reference

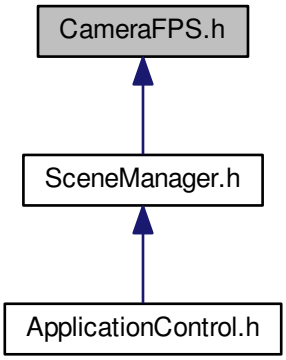
Définition de la caméra à la première personne.

```
#include "AbstractCamera.h"
```

Include dependency graph for CameraFPS.h:



This graph shows which files directly or indirectly include this file:



Classes

- class CameraFPS

  Caméra de type Freefly à deux degrés de liberté en fixant l'axe vertical.

7.8.1 Detailed Description

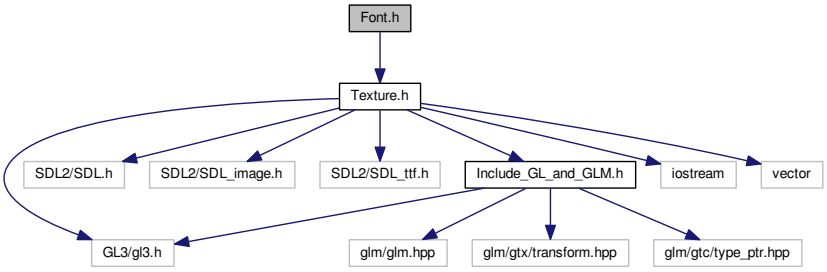
Définition de la caméra à la première personne.

7.9 Font.h File Reference

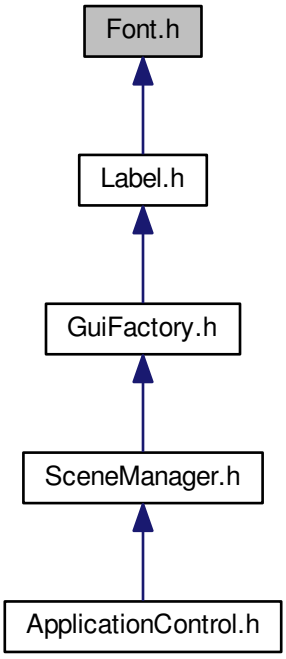
Gestion de la génération de texte.

```
#include "Texture.h"
```

Include dependency graph for Font.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Font

  Gère la conversion de texte en texture GL à partir d'une police.

7.9.1 Detailed Description

Gestion de la génération de texte.

Author

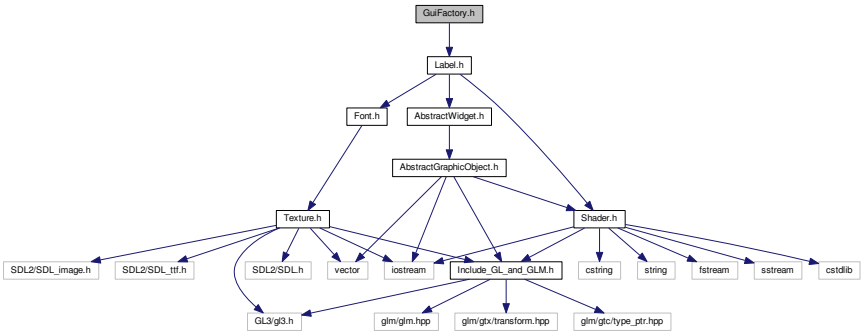
Raphaël BRESSON, Mehdi HAMMOUCHE

7.10 GuiFactory.h File Reference

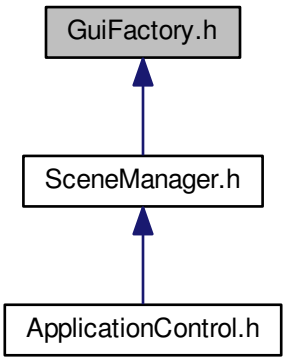
Gestion de la construction des objets 2D.

```
#include "Label.h"
```

Include dependency graph for GuiFactory.h:



This graph shows which files directly or indirectly include this file:



Classes

- class GuiFactory

  Classe de création d'objets 2D : contient les shaders d'affichage 2D.

Macros

- #define VERTEX_SHADER_GUI "Shaders/shaderGui.vert"

Vertex shader pour un objet 2D.

- #define FRAGMENT_SHADER_GUI_TEXTURE "Shaders/shaderGuiTexture.frag"

  Fragment shader pour un objet texturé

- #define FRAGMENT_SHADER_GUI_COLOR "Shaders/guiColor.vert"

  Fragment shader pour un objet non texturé (coloré)

- #define FONT_7SEGMENT "Fonts/7seg.ttf"

  Fichier de la police "afficheur 7 segments".

- #define FONT_COUNTER_STRIKE "Fonts/cs.ttf"

  Fichier de la police du jeu Counter Strike.

- #define DEFAULT_FONT_SIZE 50

  Taille par défault (résolution verticale) du texte affiché à l'écran.

7.10.1 Detailed Description

Gestion de la construction des objets 2D.

Author

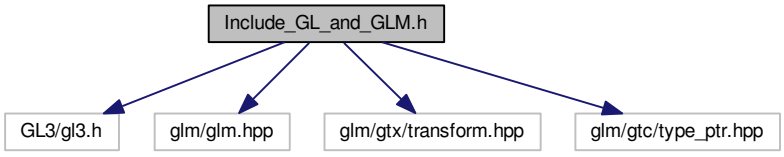
Raphaël BRESSON, Mahdi HAMMOUCHE

7.11 Include_GL_and_GLM.h File Reference

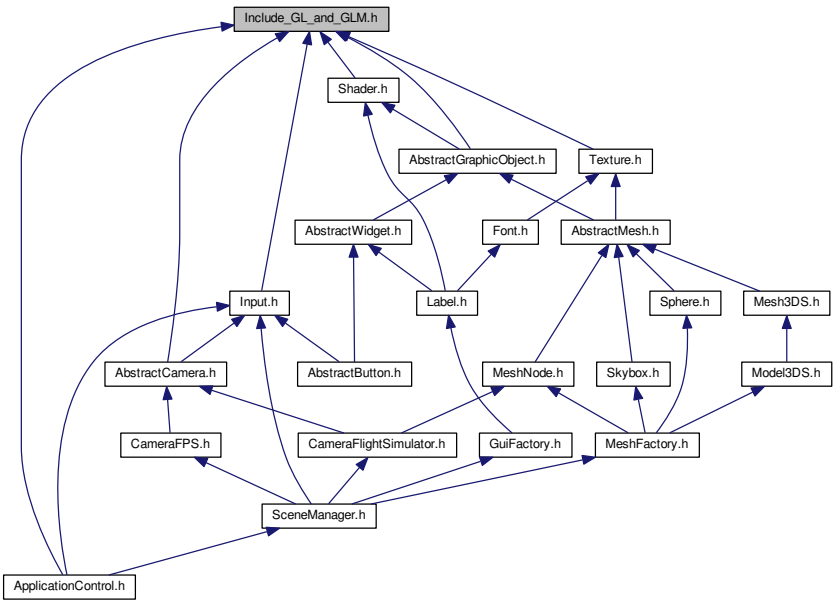
Inclusion des header de OpenGL 3 et GLM.

```
#include <GL3/gl3.h>
#include <glm/glm.hpp>
#include <glm/gtx/transform.hpp>
#include <glm/gtc/type_ptr.hpp>
```

Include dependency graph for Include_GL_and_GLM.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define GL3_PROTOTYPES 1

  Obligatoire sur Linux et Mac OS (forcer à 1)

7.11.1 Detailed Description

Inclusion des header de OpenGL 3 et GLM.

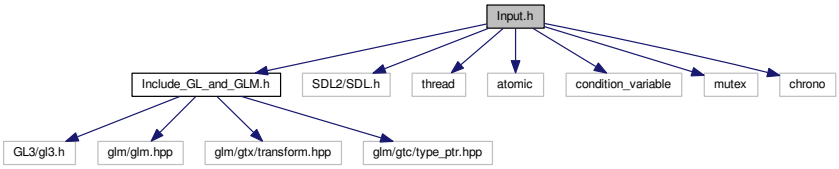
Author

Raphaël BRESSON, Mahdi HAMMOUCHE

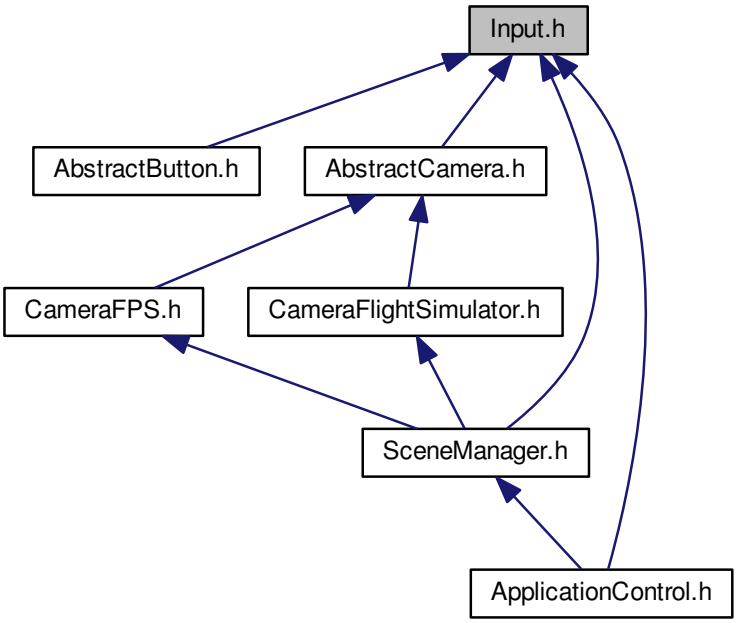
7.12 Input.h File Reference

```
#include "Include_GL_and_GLM.h"
#include <SDL2/SDL.h>
#include <thread>
#include <atomic>
#include <condition_variable>
#include <mutex>
#include <chrono>
```

Include dependency graph for Input.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Input

  gestion des évènements

7.12.1 Detailed Description

Authors

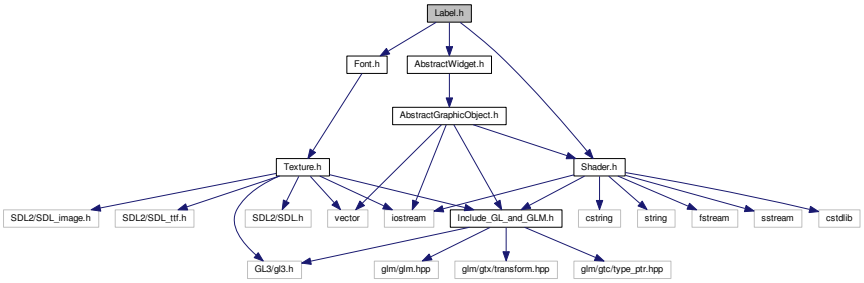
Raphaël BRESSON, Mehdi HAMMOUCHE

7.13 Label.h File Reference

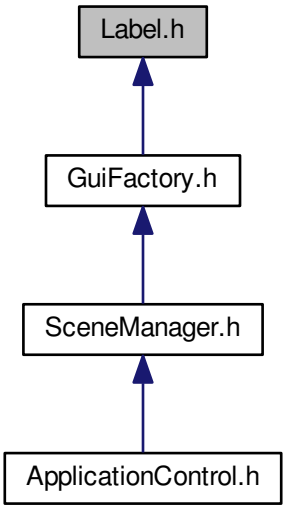
Gestion de l'affichage de texte.

```
#include "Font.h"
#include "Shader.h"
#include "AbstractWidget.h"
```

Include dependency graph for Label.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Label

  Représente un objet Texte 2D.

7.13.1 Detailed Description

Gestion de l'affichage de texte.

Author

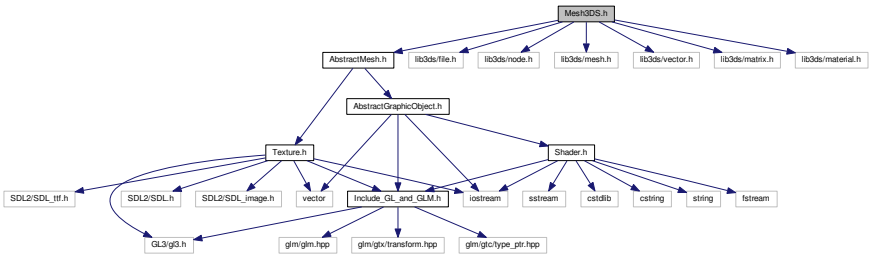
Raphaël BRESSON, Mahdi HAMMOUCHE

7.14 Mesh3DS.h File Reference

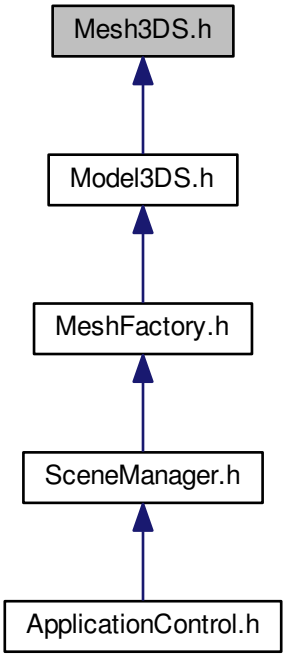
Définition d'un mesh d'un modèle 3DS.

```
#include "AbstractMesh.h"
#include <lib3ds/file.h>
#include <lib3ds/node.h>
#include <lib3ds/mesh.h>
#include <lib3ds/vector.h>
#include <lib3ds/matrix.h>
#include <lib3ds/material.h>
```

Include dependency graph for Mesh3DS.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Mesh3DS

  Mesh d'un modèle 3DS.

7.14.1 Detailed Description

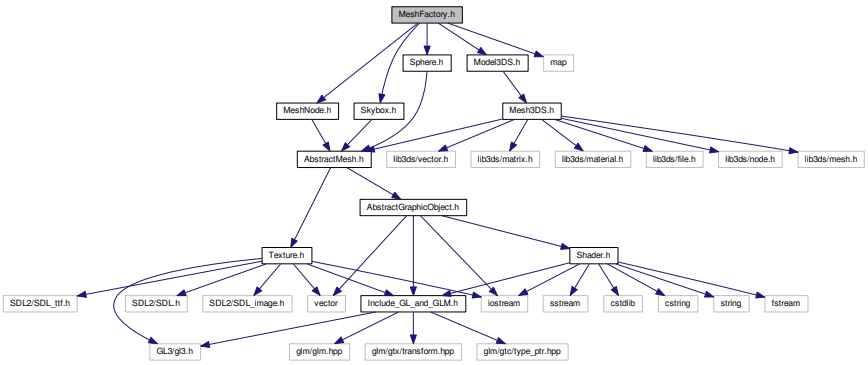
Définition d'un mesh d'un modèle 3DS.

7.15 MeshFactory.h File Reference

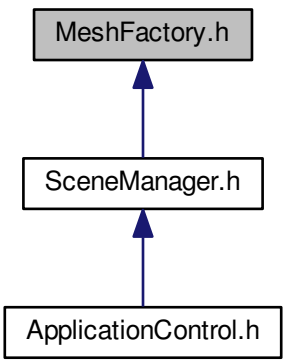
Construction d'objets 3D et gestion des shaders.

```
#include "MeshNode.h"
#include "Model3DS.h"
#include "Skybox.h"
#include "Sphere.h"
#include <map>
```

Include dependency graph for MeshFactory.h:



This graph shows which files directly or indirectly include this file:



Classes

- class MeshFactory

  Classe de gestion des shaders et de génération des modèles 3D.

Macros

- #define VERTEX_SHADER_3D "Shaders/shader3D.vert"

  Nom du vertex shader pour l'affichage 3D.

- #define FRAGMENT_SHADER_3D_TEXTURE_2D "Shaders/shader3DTexture2D.frag"

  Nom du fragment shader pour l'affichage 3D avec une texture 2D.

7.15.1 Detailed Description

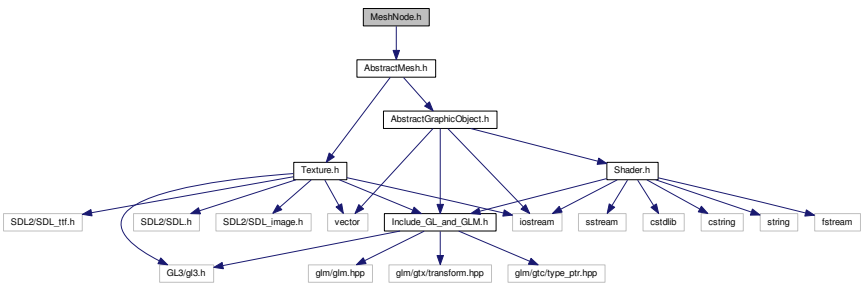
Construction d'objets 3D et gestion des shaders.

7.16 MeshNode.h File Reference

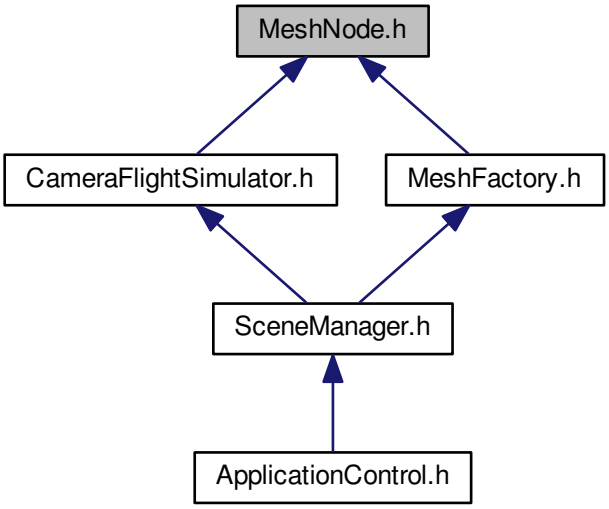
Définition de la classe de gestion de modèles 3D.

```
#include "AbstractMesh.h"
```

Include dependency graph for MeshNode.h:



This graph shows which files directly or indirectly include this file:



Classes

- class MeshNode

  Classe de gestion de modèles 3D.

7.16.1 Detailed Description

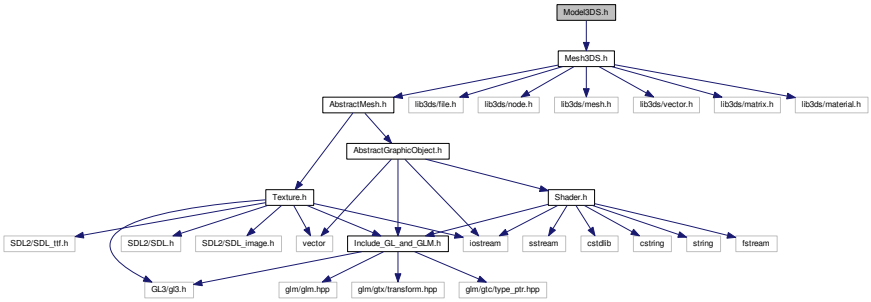
Définition de la classe de gestion de modèles 3D.

7.17 Model3DS.h File Reference

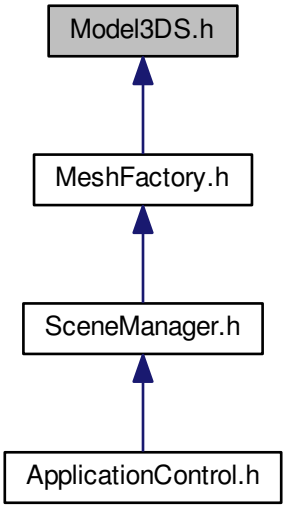
Importation d'un modèle 3D depuis le format 3DS MAX (sans les animations)

```
#include "Mesh3DS.h"
```

Include dependency graph for Model3DS.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Model3DS

  Gestion des modèle 3DS.

7.17.1 Detailed Description

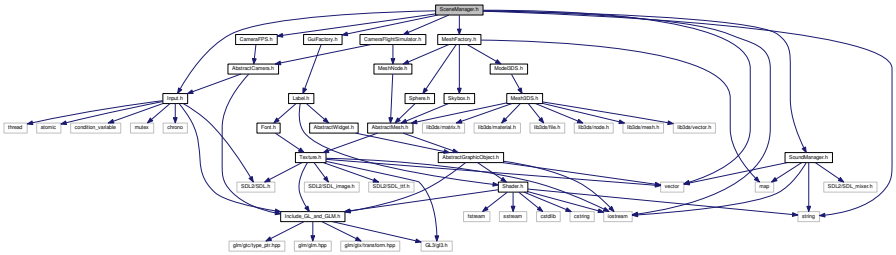
Importation d'un modèle 3D depuis le format 3DS MAX (sans les animations)

7.18 SceneManager.h File Reference

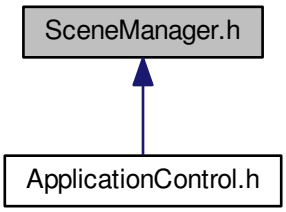
Gestion des modèles et de l'affichage.

```
#include <iostream>
#include <vector>
#include <string>
#include "CameraFPS.h"
#include "CameraFlightSimulator.h"
#include "Input.h"
#include "MeshFactory.h"
#include "GuiFactory.h"
#include "SoundManager.h"
```

Include dependency graph for SceneManager.h:



This graph shows which files directly or indirectly include this file:



Classes

- class SceneManager

  Gestionnaire des modèle, des caméras (pour le 3D) et de l'affichage.

Macros

- #define FPS_LIMIT 60

  Limitation de la fréquence de rafraichissement à 60 FPS.

7.18.1 Detailed Description

Gestion des modèles et de l'affichage.

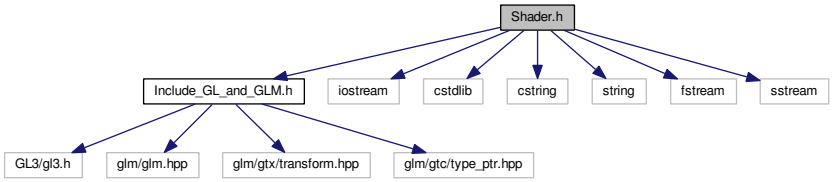
Raphaël BRESSON, Mahdi HAMMOUCHE

7.19 Shader.h File Reference

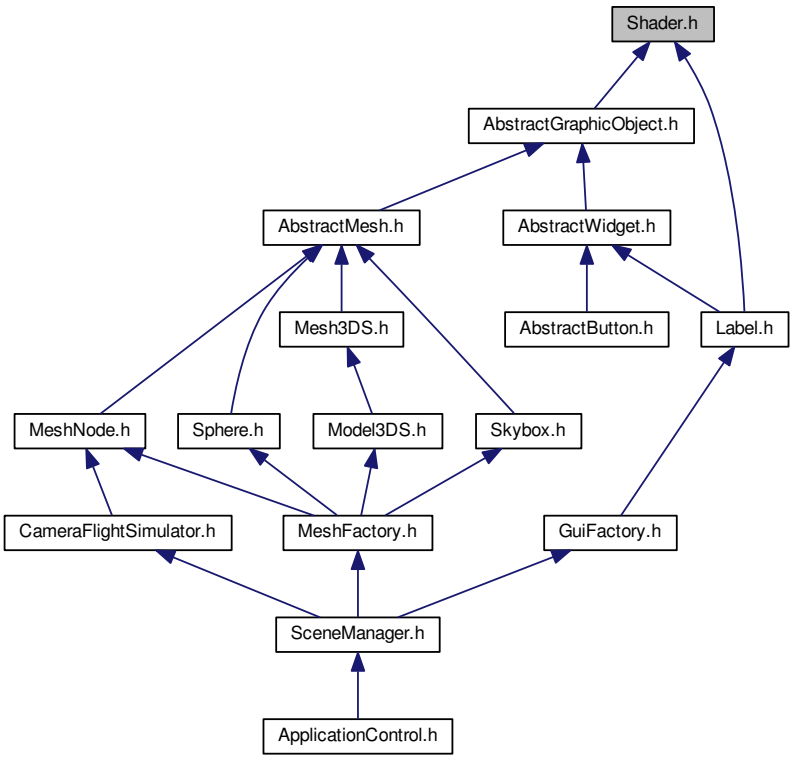
Gestion des Shaders (Programmes pour le GPU)

```
#include "Include_GL_and_GLM.h"
#include <iostream>
#include <cstdlib>
#include <cstring>
#include <string>
#include <fstream>
#include <sstream>
```

Include dependency graph for Shader.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Shader

  Classe de gestion de programmes shaders (compilation, édition de liens, contrôle et destruction)

7.19.1 Detailed Description

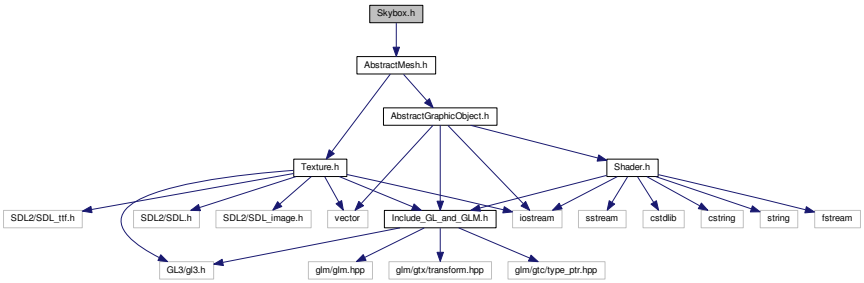
Gestion des Shaders (Programmes pour le GPU)

7.20 Skybox.h File Reference

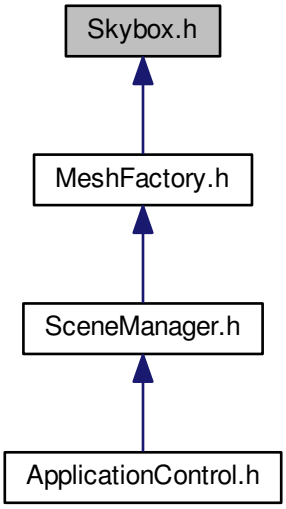
Simulation d'environnement via skybox monotexturée.

```
#include "AbstractMesh.h"
```

Include dependency graph for Skybox.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Skybox

  Cube avec textures de ciel plaquées.

7.20.1 Detailed Description

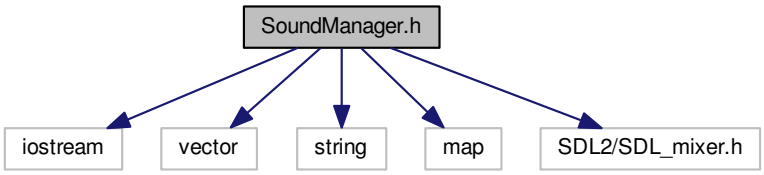
Simulation d'environnement via skybox monotexturée.

7.21 SoundManager.h File Reference

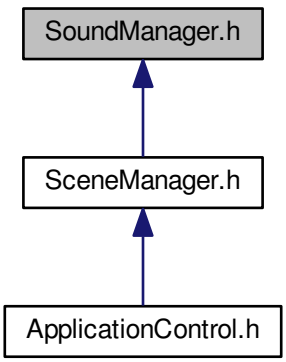
Définition du gestionnaire de sons.

```
#include <iostream>
#include <vector>
#include <string>
#include <map>
#include <SDL2/SDL_mixer.h>
```

Include dependency graph for SoundManager.h:



This graph shows which files directly or indirectly include this file:



Classes

- class SoundManager

  Gestionnaire de sons.

7.21.1 Detailed Description

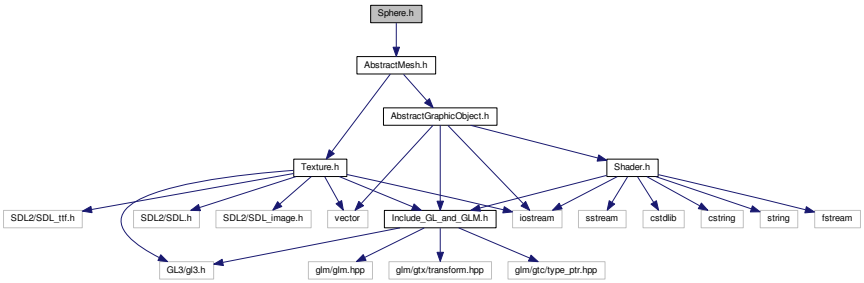
Définition du gestionnaire de sons.

7.22 Sphere.h File Reference

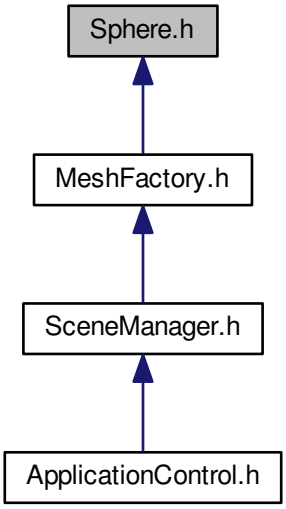
Implémentation d'une sphère texturée.

```
#include "AbstractMesh.h"
```

Include dependency graph for Sphere.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Sphere

  Définition d'une sphère texturée.

7.22.1 Detailed Description

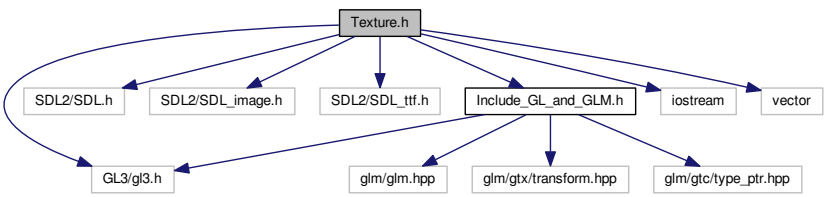
Implémentation d'une sphère texturée.

7.23 Texture.h File Reference

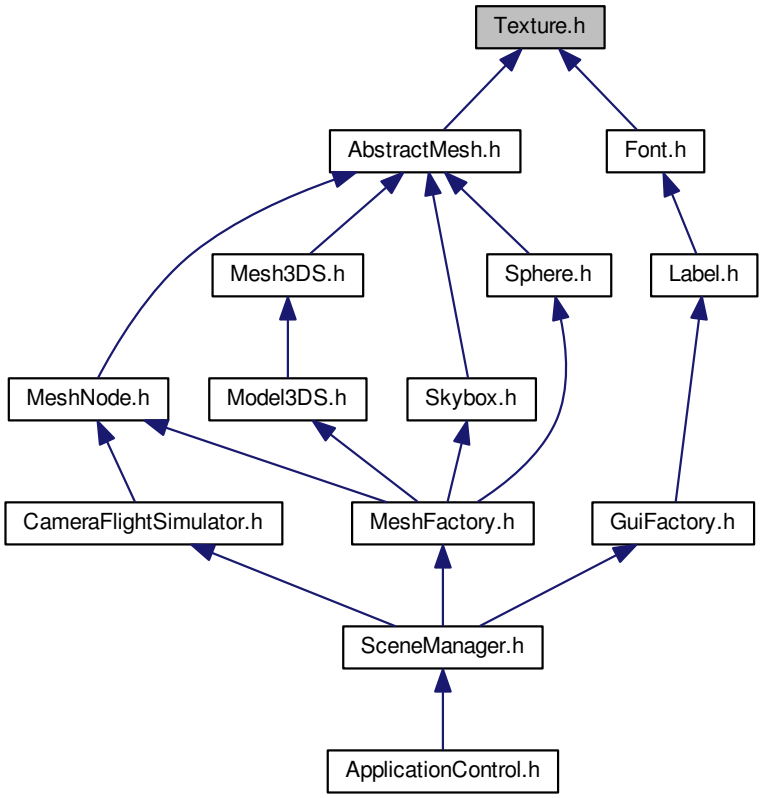
Gestion des textures.

```
#include <GL3/gl3.h>
#include <SDL2/SDL.h>
#include <SDL2/SDL_image.h>
#include <SDL2/SDL_ttf.h>
#include "Include_GL_and_GLM.h"
#include <iostream>
#include <vector>
```

Include dependency graph for Texture.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Texture

  Gère les textures OpenGL et leur importation via une image ou une police.

Macros

- #define **GL3_PROTOTYPES** 1

7.23.1 Detailed Description

Gestion des textures.

Author

Raphaël BRESSON, Mehdi HAMMOUCHE

Index