My Project

Generated by Doxygen 1.8.1.2

Mon Jan 18 2016 22:14:03

Contents

Chapter 1

Class Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

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

Chapter 4

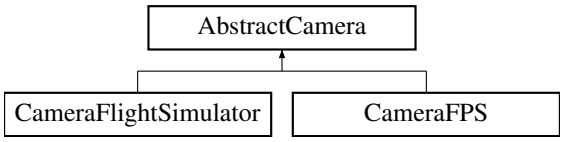
Class Documentation

4.1 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.

4.1.1 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:

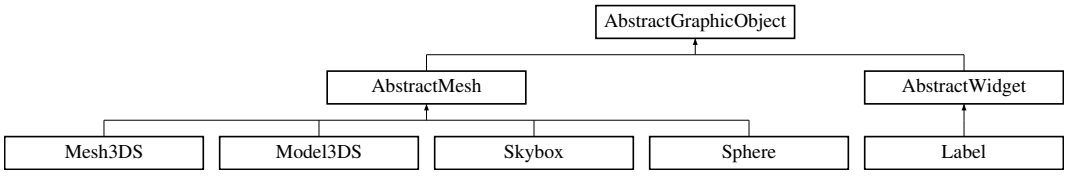
- AbstractCamera.h

4.2 AbstractGraphicObject Class Reference

Classe abstraite définissant un type polymorphe pour les objets graphiques.

`#include <AbstractGraphicObject.h>`

Inheritance 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.

4.2.1 Detailed Description

Classe abstraite définissant un type polymorphe pour les objets graphiques.

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

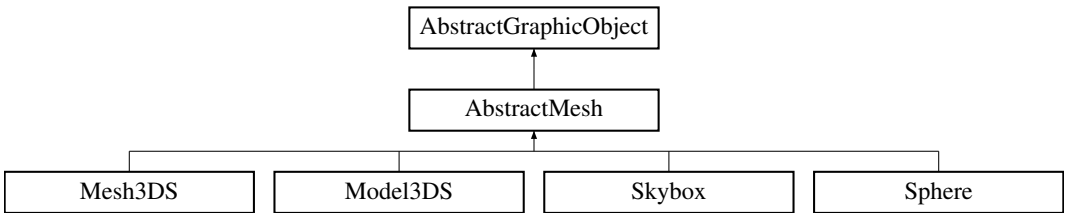
- AbstractGraphicObject.h

4.3 AbstractMesh Class Reference

Type polymorphe pour les modèles 3D.

`#include <AbstractMesh.h>`

Inheritance 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.

4.3.1 Detailed Description

Type polymorphe pour les modèles 3D.

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

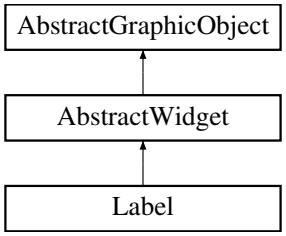
- AbstractMesh.h
- AbstractMesh.cpp

4.4 AbstractWidget Class Reference

Classe abstraite mère de tous les objets 2D.

`#include <AbstractWidget.h>`

Inheritance 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.

4.4.1 Detailed Description

Classe abstraite mère de tous les objets 2D.

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

- AbstractWidget.h

4.5 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.

4.5.1 Detailed Description

Classe repésentant globalement le programme graphique.

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

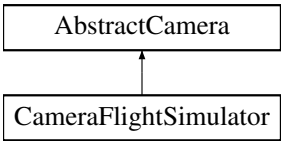
- ApplicationControl.h
- ApplicationControl.cpp

4.6 CameraFlightSimulator Class Reference

Implémentation de la caméra de simulateur de vol.

`#include <CameraFlightSimulator.h>`

Inheritance 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

4.6.1 Detailed Description

Implémentation de la caméra de simulateur de vol.

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

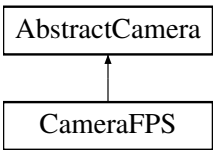
- CameraFlightSimulator.h
- CameraFlightSimulator.cpp

4.7 CameraFPS Class Reference

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

`#include <CameraFPS.h>`

Inheritance 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.

4.7.1 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:

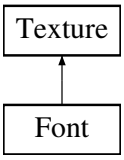
- CameraFPS.h
- CameraFPS.cpp

4.8 Font Class Reference

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

`#include <Font.h>`

Inheritance 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

4.8.1 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

4.9 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)

4.9.1 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

4.10 Input Class Reference

gestion des évènements

`#include <Input.h>`

Public Member Functions

- Input ()

  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.
- bool terminer () const

  Renvoie true si le programme doit se terminer.

4.10.1 Detailed Description

gestion des évênements

4.10.2 Constructor & Destructor Documentation

4.10.2.1 Input::Input ( )

Constructeur.

Destructeur.

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

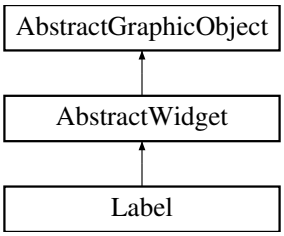
- Input.h
- Input.cpp

4.11 Label Class Reference

Représente un objet Texte 2D.

`#include <Label.h>`

Inheritance 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.

4.11.1 Detailed Description

Représente un objet Texte 2D.

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

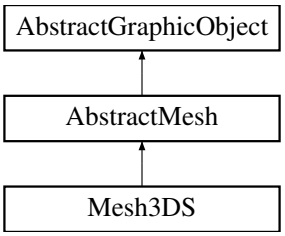
- Label.h
- Label.cpp

4.12 Mesh3DS Class Reference

Mesh d'un modèle 3DS.

`#include <Mesh3DS.h>`

Inheritance 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

4.12.1 Detailed Description

Mesh d'un modèle 3DS.

4.12.2 Member Function Documentation

4.12.2.1 void Mesh3DS::draw ( )

Affichage.

ne fait rien

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

- Mesh3DS.h
- Mesh3DS.cpp

4.13 MeshFactory Class Reference

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

`#include <MeshFactory.h>`

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)

4.13.1 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

4.14 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

4.14.1 Detailed Description

Classe de gestion de modèles 3D.

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

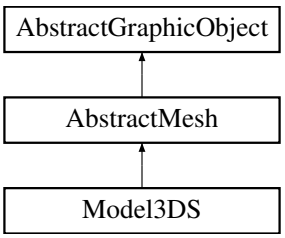
- MeshNode.h
- MeshNode.cpp

4.15 Model3DS Class Reference

Gestion des modèle 3DS.

`#include <Model3DS.h>`

Inheritance 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

4.15.1 Detailed Description

Gestion des modèle 3DS.

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

- Model3DS.h
- Model3DS.cpp

4.16 SceneManager Class Reference

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

`#include <SceneManager.h>`

Public Member Functions

- SceneManager ()

  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.

4.16.1 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

4.17 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** ()

4.17.1 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:

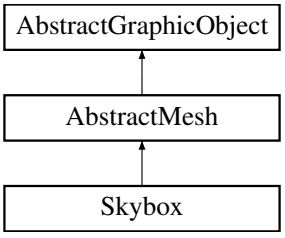
- Shader.h
- Shader.cpp

4.18 Skybox Class Reference

Cube avec textures de ciel plaquées.

`#include <Skybox.h>`

Inheritance 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

4.18.1 Detailed Description

Cube avec textures de ciel plaquées.

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

- Skybox.h
- Skybox.cpp

4.19 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.

4.19.1 Detailed Description

Gestionnaire de sons.

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

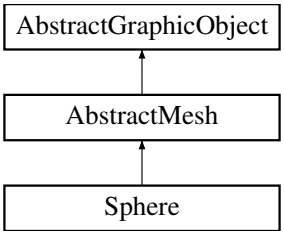
- SoundManager.h
- SoundManager.cpp

4.20 Sphere Class Reference

Définition d'une sphère texturée.

`#include <Sphere.h>`

Inheritance 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

4.20.1 Detailed Description

Définition d'une sphère texturée.

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

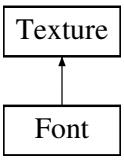
- Sphere.h
- Sphere.cpp

4.21 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.

4.21.1 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 5

File Documentation

5.1 AbstractCamera.h File Reference

Définit un type polymorphe pour les caméras.

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

Classes

- class AbstractCamera

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

5.1.1 Detailed Description

Définit un type polymorphe pour les caméras.

5.2 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"
```

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)

5.2.1 Detailed Description

Définit un type polymorphe pour les objets graphiques.

Author

Raphaël BRESSON, Mahdi HAMMOUCHE

5.3 AbstractMesh.h File Reference

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

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

Classes

- class AbstractMesh

  Type polymorphe pour les modèles 3D.

5.3.1 Detailed Description

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

5.4 AbstractWidget.h File Reference

Définit un type polymorphe pour les objets 2D.

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

Classes

- class AbstractWidget

  Classe abstraite mère de tous les objets 2D.

5.4.1 Detailed Description

Définit un type polymorphe pour les objets 2D.

5.5 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"
```

Classes

- class ApplicationControl

  Classe repésentant globalement le programme graphique.

5.5.1 Detailed Description

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

Author

Raphaël BRESSON, Mahdi HAMMOUCHE

5.6 CameraFlightSimulator.h File Reference

Implémentation de la caméra de simulateur de vol.

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

Classes

- class CameraFlightSimulator

  Implémentation de la caméra de simulateur de vol.

5.6.1 Detailed Description

Implémentation de la caméra de simulateur de vol.

5.7 CameraFPS.h File Reference

Définition de la caméra à la première personne.

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

Classes

- class CameraFPS

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

5.7.1 Detailed Description

Définition de la caméra à la première personne.

5.8 Font.h File Reference

Gestion de la génération de texte.

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

Classes

- class Font

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

5.8.1 Detailed Description

Gestion de la génération de texte.

Author

Raphaël BRESSON, Mehdi HAMMOUCHE

5.9 GuiFactory.h File Reference

Gestion de la construction des objets 2D.

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

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.

5.9.1 Detailed Description

Gestion de la construction des objets 2D.

Author

Raphaël BRESSON, Mahdi HAMMOUCHE

5.10 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>
```

Macros

- #define GL3_PROTOTYPES 1

  Obligatoire sur Linux et Mac OS (forcer à 1)

5.10.1 Detailed Description

Inclusion des header de OpenGL 3 et GLM.

Author

Raphaël BRESSON, Mahdi HAMMOUCHE

5.11 Input.h File Reference

```
#include <SDL2/SDL.h>
```

Classes

- class Input

  gestion des évênements

5.11.1 Detailed Description

Authors

Raphaël BRESSON, Mehdi HAMMOUCHE

5.12 Label.h File Reference

Gestion de l'affichage de texte.

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

Classes

- class Label

  Représente un objet Texte 2D.

5.12.1 Detailed Description

Gestion de l'affichage de texte.

Author

Raphaël BRESSON, Mahdi HAMMOUCHE

5.13 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>
```

Classes

- class Mesh3DS

  Mesh d'un modèle 3DS.

5.13.1 Detailed Description

Définition d'un mesh d'un modèle 3DS.

5.14 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>
```

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.

5.14.1 Detailed Description

Construction d'objets 3D et gestion des shaders.

5.15 MeshNode.h File Reference

Définition de la classe de gestion de modèles 3D.

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

Classes

- class MeshNode

  Classe de gestion de modèles 3D.

5.15.1 Detailed Description

Définition de la classe de gestion de modèles 3D.

5.16 Model3DS.h File Reference

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

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

Classes

- class Model3DS

  Gestion des modèle 3DS.

5.16.1 Detailed Description

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

5.17 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"
```

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.

5.17.1 Detailed Description

Gestion des modèles et de l'affichage. Raphaël BRESSON, Mahdi HAMMOUCHE

5.18 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>
```

Classes

- class Shader

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

5.18.1 Detailed Description

Gestion des Shaders (Programmes pour le GPU)

5.19 Skybox.h File Reference

Simulation d'environnement via skybox monotexturée.

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

Classes

- class Skybox

  Cube avec textures de ciel plaquées.

5.19.1 Detailed Description

Simulation d'environnement via skybox monotexturée.

5.20 SoundManager.h File Reference

Définition du gestionnaire de sons.

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

Classes

- class SoundManager

  Gestionnaire de sons.

5.20.1 Detailed Description

Définition du gestionnaire de sons.

5.21 Sphere.h File Reference

Implémentation d'une sphère texturée.

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

Classes

- class Sphere

  Définition d'une sphère texturée.

5.21.1 Detailed Description

Implémentation d'une sphère texturée.

5.22 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>
```

Classes

- class Texture

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

Macros

- #define **GL3_PROTOTYPES** 1

5.22.1 Detailed Description

Gestion des textures.

Author

Raphaël BRESSON, Mehdi HAMMOUCHE

Index