READYDEV

[GitHub](https://github.com/gkesse) | [LinkedIn](https://www.linkedin.com/in/tia-gerard-kesse/) | [SiteWeb](http://readydev.freeboxos.fr/)

Par Gérard KESSE

Plateforme de Développement Continu

Comprendre la Théorie pour mieux Pratiquer

Sciences de l'Ingénieur

Cours - Tutoriels

READYDEV

[GitHub](https://github.com/gkesse) | [LinkedIn](https://www.linkedin.com/in/tia-gerard-kesse/) | [SiteWeb](http://readydev.freeboxos.fr/)

Par Gérard KESSE

J'aime, Je partage

Montez en Compétences

OpenGL

Apprendre la programmation 3D en C++ avec OpenGL

# [Auteur](#_Sommaire)

Je suis **Gérard KESSE**,

Ingénieur en Développement Informatique C/C++/Qt,

Avec à la fois des compétences en Systèmes Embarqués et en Robotique.

Formé à Polytech'Montpellier, Je suis un professionnel de conception de projets logiciel applicatif ou embarqué dans les secteurs de l'Aéronautique, de la Robotique, des Drones et de la Vision par Ordinateur. Aussi, Je reste ouvert à d'autres types de secteurs tels que l'Énergie et les Finances.

Les Sciences de l’Ingénieur sont au cœur du métier d’ingénieur. Sur le site

**ReadyDev**, la Plateforme de Développement Continu, dont j'en suis le concepteur, vous trouverez des cours et des tutoriels adaptés aux sciences de l’ingénieur.

J'aime, Je partage.

Gérard KESSE

[GitHub](https://github.com/gkesse) | [LinkedIn](https://www.linkedin.com/in/tia-gerard-kesse/) | [SiteWeb](http://readydev.freeboxos.fr/)

[](http://readydev.freeboxos.fr/)

# Sommaire

[Auteur 2](#_Toc511460584)

[Sommaire 3](#_Toc511460585)

[Introduction 4](#_Toc511460586)

[Installation sous Windows avec MinGW 4](#_Toc511460587)

[Téléchargements 4](#_Toc511460588)

[Installation de Notepad++ 5](#_Toc511460589)

[Installation de MinGW 5](#_Toc511460590)

[Installation de FreeGLUT 5](#_Toc511460591)

[Utiliser OpenGL avec SDL et GLEW 7](#_Toc511460592)

[Afficher une fenêtre 7](#_Toc511460593)

[Ajouter une couleur de fond d’écran 10](#_Toc511460594)

[Afficher un objet dans la fenêtre 13](#_Toc511460595)

[Ajouter une caméra à la scène 3D 17](#_Toc511460596)

[Ajouter de la lumière à une scène 3D 20](#_Toc511460597)

# [Introduction](#_Sommaire)

Le C++ est un langage de programmation orienté objet. OpenGL est une bibliothèque de création d’applications 3D. Le but de ce tutoriel est de vous apprendre la programmation 3D en C++ avec OpenGL.

**Prérequis :**

Apprendre à compiler un projet C++ avec QMake.

# [Installation sous Windows avec MinGW](#_Sommaire)

OpenGL (Open Graphics Library) est un ensemble normalisé de fonctions de calcul d'images 2D ou 3D disponible sur de nombreuses plateformes où elle est utilisée pour des applications qui vont du jeu vidéo jusqu'à la CAO en passant par la modélisation.

OpenGL permet à un programme de déclarer la géométrie d'objets sous forme de points, de vecteurs, de polygones, de bitmaps et de textures. OpenGL effectue ensuite des calculs de projection en vue de déterminer l'image à l'écran, en tenant compte de la distance, de l'orientation, des ombres, de la transparence et du cadrage.

Dans ce tutoriel, nous utilisons la bibliothèque OpenGL avec la bibliothèque SDL, l’utilitaire GLEW, et l’IDE Qt Creator avec le compilateur C++ fourni par MinGW.

## [Téléchargements](#_Sommaire)

**Notepad++ :**

<https://notepad-plus-plus.org/fr/>

**MinGW :**

<http://mingw.org/>

**Qt Creator :**

<https://download.qt.io/archive/qt/>

**CMake :**

<https://cmake.org/>

**SDL :**

<https://www.libsdl.org/>

**GLEW :**

<http://glew.sourceforge.net/>

## [Installation de Notepad++](#_Sommaire)

**Plugins Notepad++ :**

TextFX

NppExport

## [Installation de MinGW](#_Sommaire)

**Packages MinGW :**

mingw32-base

mingw32-gcc-g++

# [Utiliser OpenGL avec SDL et GLEW](#_Sommaire)

## [Afficher une fenêtre](#_Sommaire)

**Objectif :**

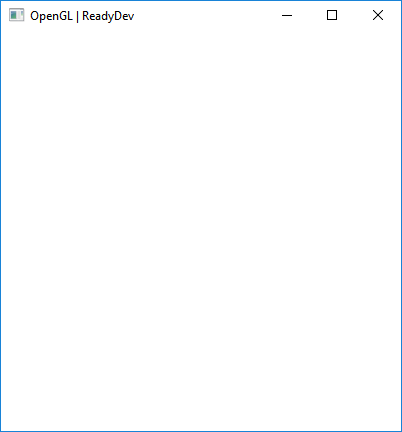
Afficher une fenêtre.

**Implémentation :**

Créer un gestionnaire de fenêtre (GWindow).

Afficher la fenêtre (show()).

**Résultat :**



**Dossier projet :**

src/main.cpp

win/GWindow.h

win/GWindow.cpp

win/bin/

win/build/

**Programme principale :**

//===============================================

#include "GWindow.h"

//===============================================

int main**(**int argc**,** char**\*\*** argv**)** **{**

GWindow**::**Instance**()->**show**();**

**return** 0**;**

**}**

//===============================================

**Affichage de la fenêtre :**

//===============================================

void GWindow**::**show**()** **{**

SDL\_Init**(**SDL\_INIT\_VIDEO**);**

SDL\_GL\_SetAttribute**(**SDL\_GL\_CONTEXT\_MAJOR\_VERSION**,** 3**);**

SDL\_GL\_SetAttribute**(**SDL\_GL\_CONTEXT\_MINOR\_VERSION**,** 1**);**

SDL\_GL\_SetAttribute**(**SDL\_GL\_DOUBLEBUFFER**,** 1**);**

SDL\_GL\_SetAttribute**(**SDL\_GL\_DEPTH\_SIZE**,** 24**);**

m\_window **=** SDL\_CreateWindow**(**m\_title**.**toStdString**().**c\_str**(),** m\_x**,** m\_y**,** m\_w**,** m\_h**,** SDL\_WINDOW\_SHOWN | SDL\_WINDOW\_OPENGL**);**

SDL\_GLContext m\_context **=** SDL\_GL\_CreateContext**(**m\_window**);**

SDL\_Event m\_event**;**

**while(**1**)** **{**

SDL\_WaitEvent**(&**m\_event**);**

**if(**m\_event**.**window**.**event **==** SDL\_WINDOWEVENT\_CLOSE**)** **break;**

**}**

SDL\_GL\_DeleteContext**(**m\_context**);**

SDL\_DestroyWindow**(**m\_window**);**

SDL\_Quit**();**

**}**

//===============================================

**Initialisation des paramètres de la fenêtre :**

//===============================================

GWindow**::**GWindow**()** **{**

m\_x **=** SDL\_WINDOWPOS\_CENTERED**;**

m\_y **=** SDL\_WINDOWPOS\_CENTERED**;**

m\_w **=** 400**;**

m\_h **=** 400**;**

m\_title **=** "OpenGL | ReadyDev"**;**

m\_window **=** 0**;**

**}**

//===============================================

## [Ajouter une couleur de fond d’écran](#_Sommaire)

**Objectif :**

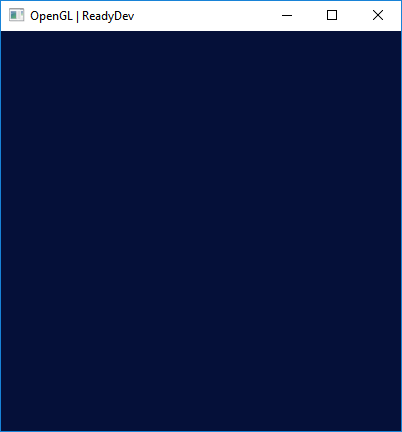
Ajouter une couleur de fond d’écran à une fenêtre.

**Implémentation :**

Créer un gestionnaire de fenêtre (GWindow).

Ajouter une couleur de fond d’écran (setBackground()).

**Résultat :**



**Dossier projet :**

src/main.cpp

src/GWindow.h

src/GWindow.cpp

win/bin/

win/build/

**Programme principal :**

//===============================================

#include "GWindow.h"

//===============================================

int main**(**int argc**,** char**\*\*** argv**)** **{**

GWindow**::**Instance**()->**show**();**

**return** 0**;**

**}**

//===============================================

**Affichage de la fenêtre :**

//===============================================

void GWindow**::**show**()** **{**

SDL\_Init**(**SDL\_INIT\_VIDEO**);**

SDL\_GL\_SetAttribute**(**SDL\_GL\_CONTEXT\_MAJOR\_VERSION**,** 3**);**

SDL\_GL\_SetAttribute**(**SDL\_GL\_CONTEXT\_MINOR\_VERSION**,** 1**);**

SDL\_GL\_SetAttribute**(**SDL\_GL\_DOUBLEBUFFER**,** 1**);**

SDL\_GL\_SetAttribute**(**SDL\_GL\_DEPTH\_SIZE**,** 24**);**

m\_window **=** SDL\_CreateWindow**(**m\_title**.**toStdString**().**c\_str**(),** m\_x**,** m\_y**,** m\_w**,** m\_h**,** SDL\_WINDOW\_SHOWN **|** SDL\_WINDOW\_OPENGL**);**

SDL\_GLContext m\_context **=** SDL\_GL\_CreateContext**(**m\_window**);**

glewInit**();**

SDL\_Event m\_event**;**

**while(**1**)** **{**

SDL\_WaitEvent**(&**m\_event**);**

**if(**m\_event**.**window**.**event **==** SDL\_WINDOWEVENT\_CLOSE**)** **break;**

setBackground**();**

glClear**(**GL\_COLOR\_BUFFER\_BIT**);**

SDL\_GL\_SwapWindow**(**m\_window**);**

**}**

SDL\_GL\_DeleteContext**(**m\_context**);**

SDL\_DestroyWindow**(**m\_window**);**

SDL\_Quit**();**

**}**

//===============================================

**Ajout d’une couleur de fond d’écran :**

//===============================================

void GWindow**::**setBackground**()** **{**

float m\_red **=** 5.0**/**255.0**;**

float m\_green **=** 16.0**/**255.0**;**

float m\_blue **=** 57.0**/**255.0**;**

float m\_alpha **=** 255.0**/**255.0**;**

glClearColor**(**m\_red**,** m\_green**,** m\_blue**,** m\_alpha**);**

**}**

//===============================================

## [Afficher un objet dans la fenêtre](#_Sommaire)

**Objectif :**

Afficher un objet dans la fenêtre.

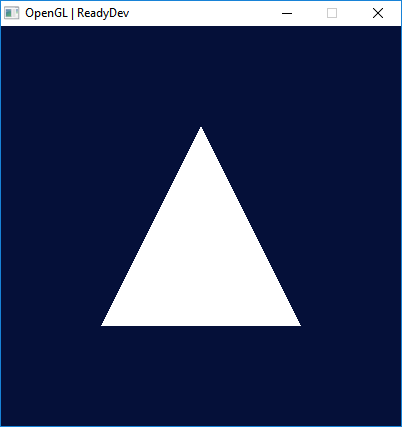
**Implémentation :**

Créer un gestionnaire de fenêtre (GWindow).

Créer un gestionnaire de dessin (GDraw).

Dessiner un objet dans la fenêtre (draw()).

**Résultat :**



**Dossier projet :**

src/main.cpp

src/GWindow.h

src/GWindow.cpp

win/bin/

win/build/

**Programme principal :**

//===============================================

#include "GWindow.h"

//===============================================

int main**(**int argc**,** char**\*\*** argv**)** **{**

GWindow**::**Instance**()->**show**();**

**return** 0**;**

**}**

//===============================================

**Affichage de la fenêtre :**

//===============================================

void GWindow**::**show**()** **{**

SDL\_Init**(**SDL\_INIT\_VIDEO**);**

SDL\_GL\_SetAttribute**(**SDL\_GL\_CONTEXT\_MAJOR\_VERSION**,** 3**);**

SDL\_GL\_SetAttribute**(**SDL\_GL\_CONTEXT\_MINOR\_VERSION**,** 1**);**

SDL\_GL\_SetAttribute**(**SDL\_GL\_DOUBLEBUFFER**,** 1**);**

SDL\_GL\_SetAttribute**(**SDL\_GL\_DEPTH\_SIZE**,** 24**);**

m\_window **=** SDL\_CreateWindow**(**m\_title**.**toStdString**().**c\_str**(),** m\_x**,** m\_y**,** m\_w**,** m\_h**,** SDL\_WINDOW\_SHOWN **|** SDL\_WINDOW\_OPENGL**);**

SDL\_GLContext m\_context **=** SDL\_GL\_CreateContext**(**m\_window**);**

glewInit**();**

GDraw**::**Instance**()->**setVertices**();**

SDL\_Event m\_event**;**

**while(**1**)** **{**

SDL\_WaitEvent**(&**m\_event**);**

**if(**m\_event**.**window**.**event **==** SDL\_WINDOWEVENT\_CLOSE**)** **break;**

setBackground**();**

glClear**(**GL\_COLOR\_BUFFER\_BIT**);**

GDraw**::**Instance**()->**draw**();**

SDL\_GL\_SwapWindow**(**m\_window**);**

**}**

SDL\_GL\_DeleteContext**(**m\_context**);**

SDL\_DestroyWindow**(**m\_window**);**

SDL\_Quit**();**

**}**

//===============================================

**Initialisation des vertices :**

//===============================================

void GDraw**::**setVertices**()** **{**

float m\_vertices**[]** **=** **{-**0.5**,** **-**0.5**,** 0.0**,** 0.5**,** 0.5**,** **-**0.5**};**

int m\_size **=** **sizeof(**m\_vertices**)/sizeof(**float**);**

**for(**int i **=** 0**;** i **<** m\_size**;** i**++)** **{**

m\_verticesMap**[**i**]** **=** m\_vertices**[**i**];**

**}**

**}**

//===============================================

**Dessin de l’objet de l’objet dans la fenêtre :**

//===============================================

void GDraw**::**draw**()** **{**

glVertexAttribPointer**(**0**,** 2**,** GL\_FLOAT**,** GL\_FALSE**,** 0**,** m\_verticesMap**);**

glEnableVertexAttribArray**(**0**);**

glDrawArrays**(**GL\_TRIANGLES**,** 0**,** 3**);**

glDisableVertexAttribArray**(**0**);**

**}**

//===============================================

## [Ajouter une couleur à un objet](#_Sommaire)

**Objectif :**

Ajouter une couleur à un objet.

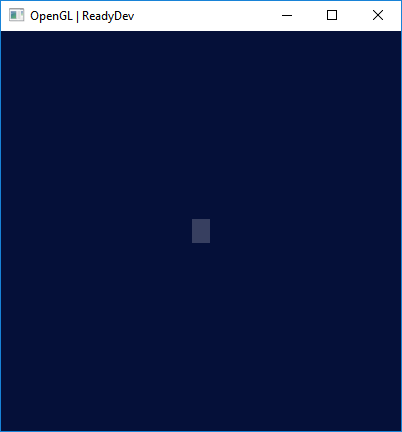
**Implémentation :**

Créer un gestionnaire de fenêtre (GWindow).

Créer un gestionnaire de shader (GShader).

Ajouter une couleur à l’objet.

**Résultat :**



**Dossier projet :**

src/main.cpp

src/GWindow.h

src/GWindow.cpp

win/bin/

win/build/

**Programme principal :**

//===============================================

#include "GWindow.h"

//===============================================

int main**(**int argc**,** char**\*\*** argv**)** **{**

GWindow**::**Instance**()->**show**(&**argc**,** argv**);**

**return** 0**;**

**}**

//===============================================

**Affichage de la fenêtre :**

//===============================================

void GWindow**::**show**(**int**\*** argc**,** char**\*\*** argv**)** **{**

glutInit**(**argc**,** argv**);**

glutInitDisplayMode**(**GLUT\_DOUBLE **|** GLUT\_RGBA **|** GLUT\_DEPTH**);**

glutInitWindowPosition**(**m\_x**,** m\_y**);**

glutInitWindowSize**(**m\_w**,** m\_h**);**

glutCreateWindow**(**m\_title**.**toStdString**().**c\_str**());**

glutSetOption **(**GLUT\_ACTION\_ON\_WINDOW\_CLOSE**,** GLUT\_ACTION\_GLUTMAINLOOP\_RETURNS**);**

setBackground**();**

glutReshapeFunc**(**ReshapeFunc**);**

glutIdleFunc**(**IdleFunc**);**

glutDisplayFunc**(**DisplayFunc**);**

glutMainLoop**();**

**}**

//===============================================

**Fonction de rappel de la tâche de fond :**

//===============================================

void GWindow**::**IdleFunc**()** **{**

glClear**(**GL\_COLOR\_BUFFER\_BIT **|** GL\_DEPTH\_BUFFER\_BIT**);**

GDraw**::**Instance**()->**draw**();**

GCamera**::**Instance**()->**update**();**

glutSwapBuffers**();**

**}**

//===============================================

**Fonction de rappel du redimensionnement de la fenêtre :**

//===============================================

void GWindow**::**ReshapeFunc**(**int w**,** int h**)** **{**

glViewport**(**0**,** 0**,** w**,** h**);**

GCamera**::**Instance**()->**setWindowSize**(**w**,** h**);**

GCamera**::**Instance**()->**update**();**

**}**

//===============================================

**Affichage du champ de vision de la camera :**

//===============================================

void GCamera**::**update**()** **{**

**if(**m\_w **==** 0 **&&** m\_h **==** 0**)** **return;**

glMatrixMode**(**GL\_PROJECTION**);**

glLoadIdentity**();**

float m\_ratio **=** **(**float**)**m\_w**/**m\_h**;**

float m\_nearPlane **=** 1.0**;**

float m\_farPlane **=** 1000.0**;**

float m\_left **=** **-**m\_ratio**\***m\_nearPlane**;**

float m\_right **=** m\_ratio**\***m\_nearPlane**;**

float m\_top **=** **-**m\_nearPlane**;**

float m\_bottom **=** m\_nearPlane**;**

float m\_zNear **=** m\_nearPlane**;**

float m\_zFar **=** m\_farPlane**;**

glFrustum**(**m\_left**,** m\_right**,** m\_top**,** m\_bottom**,** m\_zNear**,** m\_zFar**);**

glMatrixMode**(**GL\_MODELVIEW**);**

glLoadIdentity**();**

float m\_eyeX **=** 10.0**;**

float m\_eyeY **=** 5.0**;**

float m\_eyeZ **=** 0.0**;**

float m\_centerX **=** 0.0**;**

float m\_centerY **=** 0.0**;**

float m\_centerZ **=** 0.0**;**

float m\_upX **=** 0.0**;**

float m\_upY **=** 1.0**;**

float m\_upZ **=** 0.0**;**

gluLookAt**(**m\_eyeX**,** m\_eyeY**,** m\_eyeZ**,** m\_centerX**,** m\_centerY**,** m\_centerZ**,** m\_upX**,** m\_upY**,** m\_upZ**);**

**}**

//===============================================

## [Ajouter de la lumière à une scène 3D](#_Sommaire)

**Objectif :**

Ajouter de la lumière à une scène 3D.

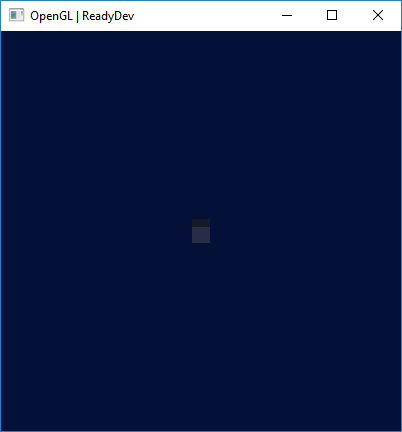
**Implémentation :**

Créer un gestionnaire de fenêtre (GWindow).

Créer un gestionnaire de lumière (GLight).

Ajouter de la lumière à la scène (initLight()).

**Résultat :**



**Dossier projet :**

src/main.cpp

src/GWindow.h

src/GWindow.cpp

win/bin/

win/build/

**Programme principal :**

//===============================================

#include "GWindow.h"

//===============================================

int main**(**int argc**,** char**\*\*** argv**)** **{**

GWindow**::**Instance**()->**show**(&**argc**,** argv**);**

**return** 0**;**

**}**

//===============================================

**Affichage de la fenêtre :**

//===============================================

void GWindow**::**show**(**int**\*** argc**,** char**\*\*** argv**)** **{**

glutInit**(**argc**,** argv**);**

glutInitDisplayMode**(**GLUT\_DOUBLE **|** GLUT\_RGBA **|** GLUT\_DEPTH**);**

glutInitWindowPosition**(**m\_x**,** m\_y**);**

glutInitWindowSize**(**m\_w**,** m\_h**);**

glutCreateWindow**(**m\_title**.**toStdString**().**c\_str**());**

glutSetOption **(**GLUT\_ACTION\_ON\_WINDOW\_CLOSE**,** GLUT\_ACTION\_GLUTMAINLOOP\_RETURNS**);**

setLight**();**

setBackground**();**

glutReshapeFunc**(**ReshapeFunc**);**

glutIdleFunc**(**IdleFunc**);**

glutDisplayFunc**(**DisplayFunc**);**

glutMainLoop**();**

**}**

//===============================================

**Ajout de la lumière à la scène 3D :**

//===============================================

void GWindow**::**setLight**()** **{**

GLight**::**Instance**()->**initLight**();**

**}**

//===============================================

**Création de la lumière ambiante, diffuse, spéculaire :**

//===============================================

void GLight**::**initLight**()** **{**

GLfloat m\_ambient**[]** **=** **{**51.0**/**255.0**,** 51.0**/**255.0**,** 51.0**/**255.0**,** 255.0**/**255.0**};**

GLfloat m\_diffuse**[]** **=** **{**255.0**/**255.0**,** 255.0**/**255.0**,** 255.0**/**255.0**,** 255.0**/**255.0**};**

GLfloat m\_specular**[]** **=** **{**255.0**/**255.0**,** 255.0**/**255.0**,** 255.0**/**255.0**,** 255.0**/**255.0**};**

GLfloat m\_position**[]** **=** **{**5.0**,** 10.0**,** 1.0**,** 0.0**};**

glLightfv**(**GL\_LIGHT0**,** GL\_AMBIENT**,** m\_ambient**);**

glLightfv**(**GL\_LIGHT0**,** GL\_DIFFUSE**,** m\_diffuse**);**

glLightfv**(**GL\_LIGHT0**,** GL\_SPECULAR**,** m\_specular**);**

glLightfv**(**GL\_LIGHT0**,** GL\_POSITION**,** m\_position**);**

glEnable**(**GL\_LIGHTING**);**

glEnable**(**GL\_LIGHT0**);**

glEnable**(**GL\_COLOR\_MATERIAL**);**

glMaterialfv**(**GL\_FRONT**,** GL\_SPECULAR**,** m\_specular**);**

glMateriali**(**GL\_FRONT**,** GL\_SHININESS**,** 15**);**

glShadeModel**(**GL\_SMOOTH**);**

glEnable**(**GL\_DEPTH\_TEST**);**

glDepthFunc**(**GL\_LESS**);**

**}**

//===============================================