Project Collision 中的渲染与工程配置

李知含

2016 年 12 月 21 日

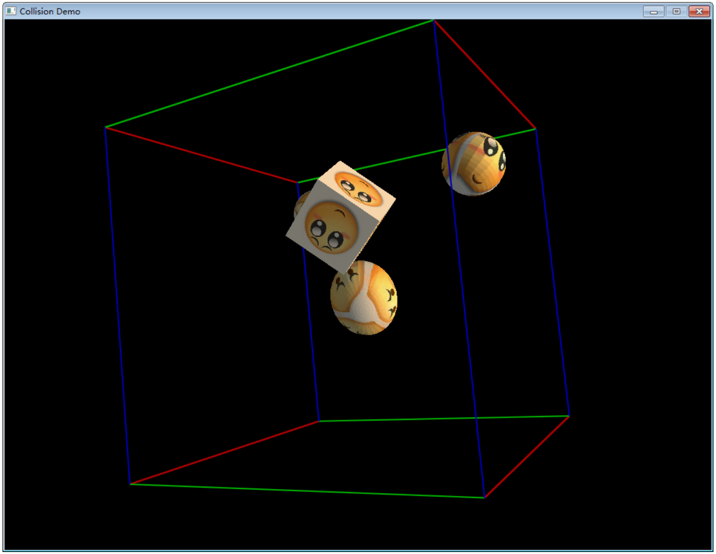
图形渲染中的几个原理

深度缓存 (Depth buffering, z-buffering)

1. 画家算法;
2. 深度缓存;
3. 投影变换, 深度值, z 值.

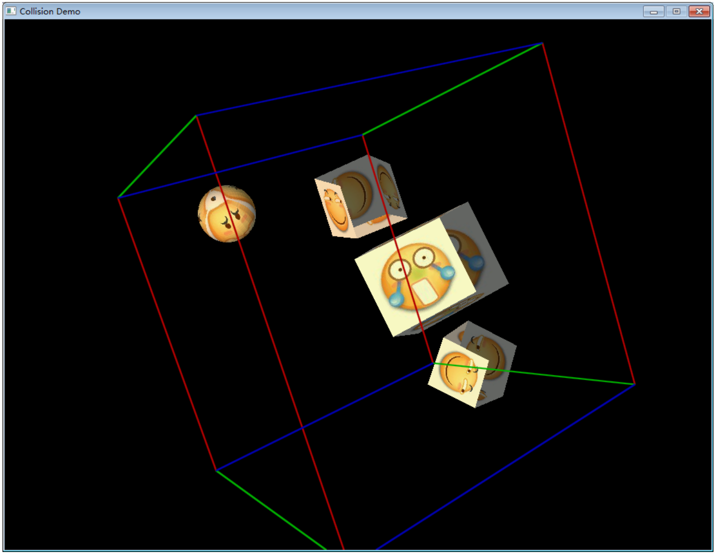
图形渲染中的几个原理

深度缓存 (Depth buffering, *z*-buffering)



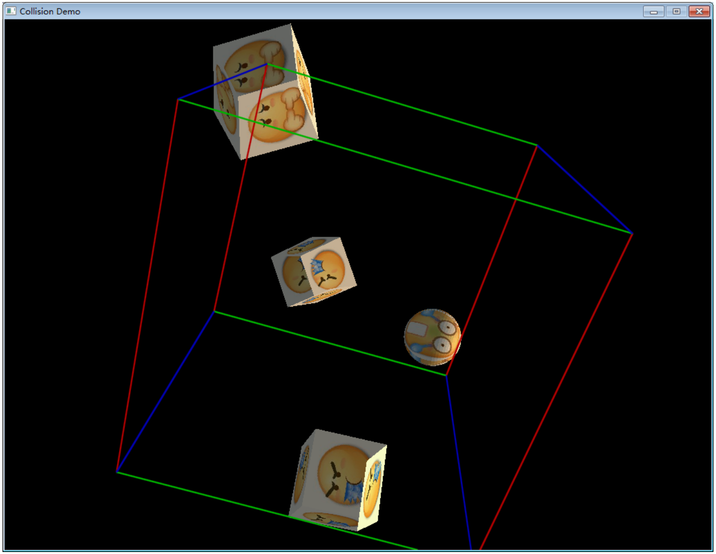
图形渲染中的几个原理

深度缓存 (Depth buffering, *z*-buffering)



图形渲染中的几个原理

深度缓存 (Depth buffering, *z*-buffering)



图形渲染中的几个原理

深度缓存 (Depth buffering, *z*-buffering)

1. `glEnable(GL_DEPTH_TEST);` 开启深度测试;
2. `glDisable(GL_DEPTH_TEST);` 禁用深度测试;
3. `glDepthFunc(GL_LESS);` 设置深度测试函数.

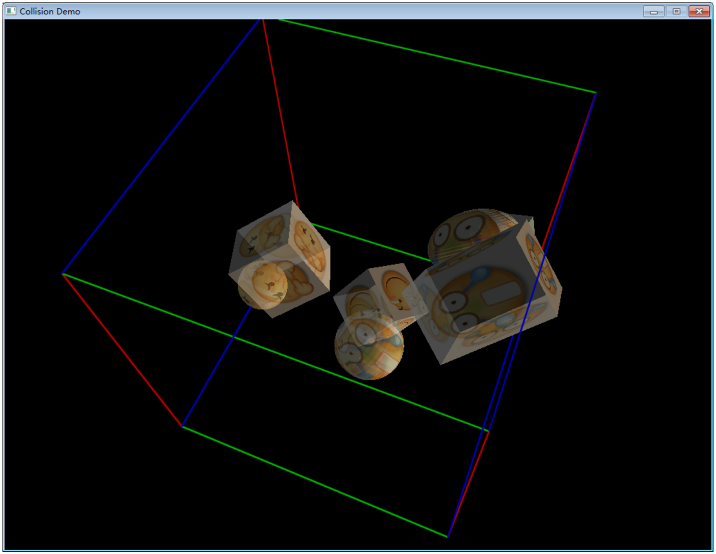
图形渲染中的几个原理

α 混合 (Alpha-blending)

1. Alpha 值;
2. 目标颜色与原颜色;
3. 混合因子;
4. $$C_i = (C_S \cdot S) + (C_D \cdot D)\,. \tag{1}$$

图形渲染中的几个原理

α 混合 (Alpha-blending)



图形渲染中的几个原理

α 混合 (Alpha-blending)

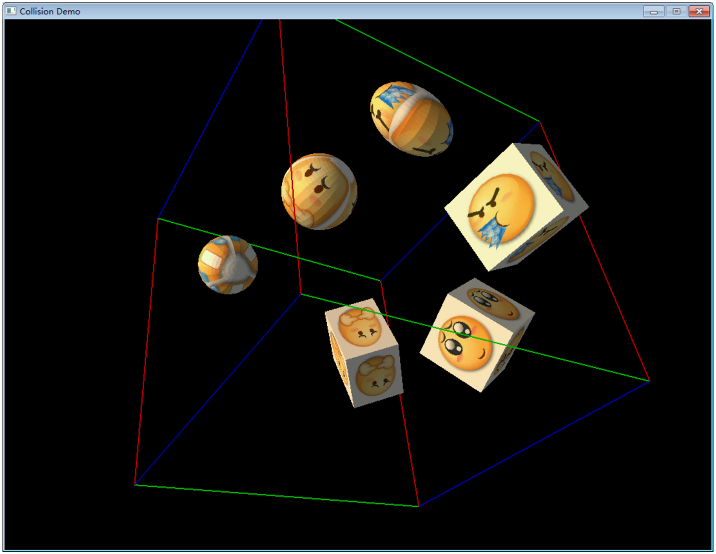
1. `glEnable(GL_BLEND);` 开启深度测试;
2. `glDisable(GL_BLEND);` 禁用深度测试;
3. `glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);` 设置深度测试函数.

图形渲染中的几个原理
抗锯齿 (Anti-aliasing)

1. 对点与直线的简单判断;
2. 超级采样抗锯齿 (SSAA);
3. 多重采样抗锯齿 (MSAA).

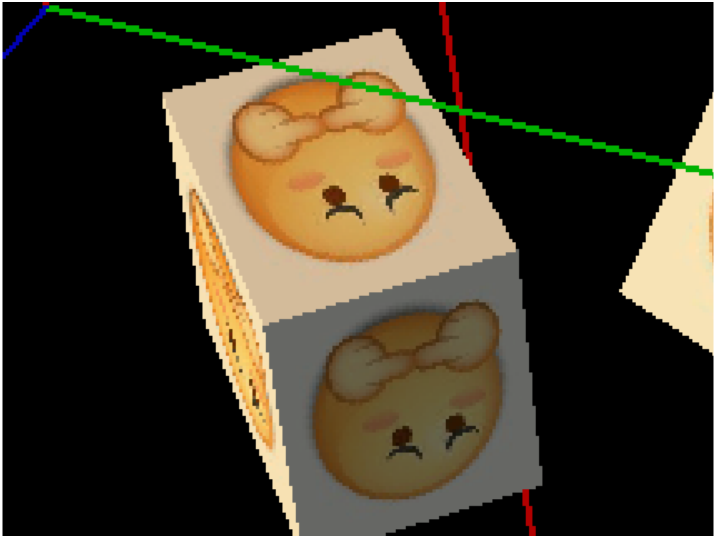
图形渲染中的几个原理

抗锯齿 (Anti-aliasing)



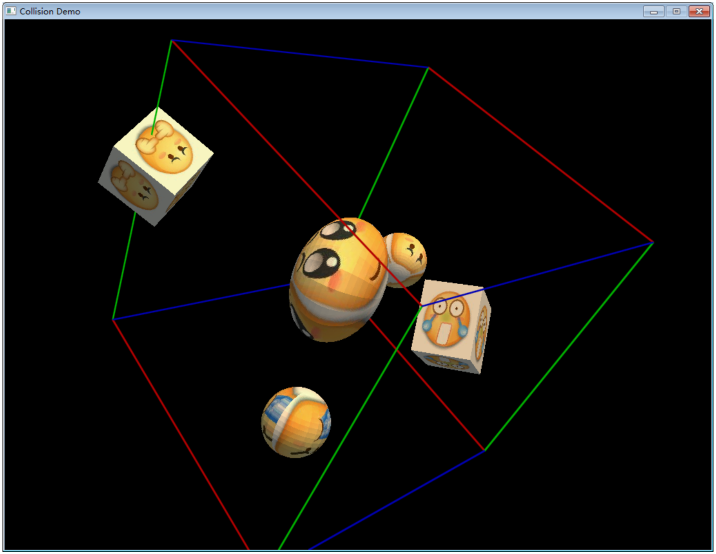
图形渲染中的几个原理

抗锯齿 (Anti-aliasing)



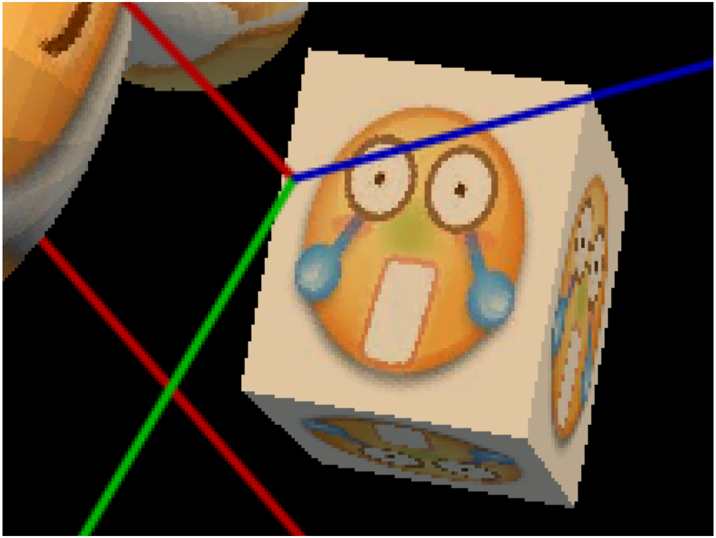
图形渲染中的几个原理

抗锯齿 (Anti-aliasing)



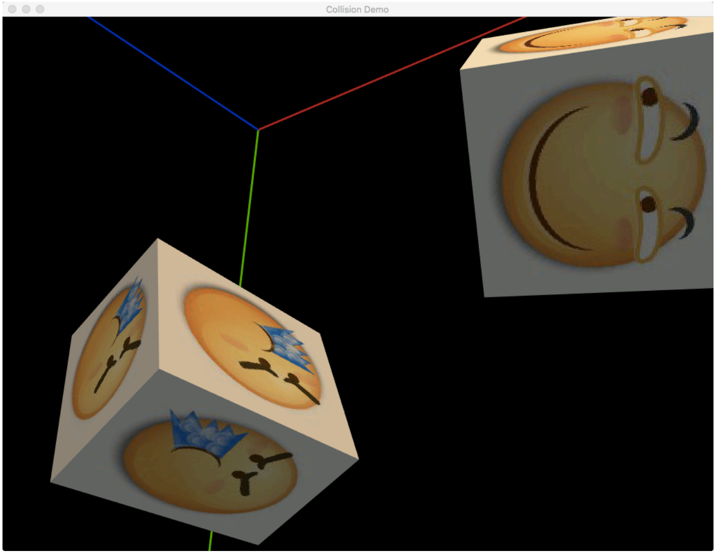
图形渲染中的几个原理

抗锯齿 (Anti-aliasing)



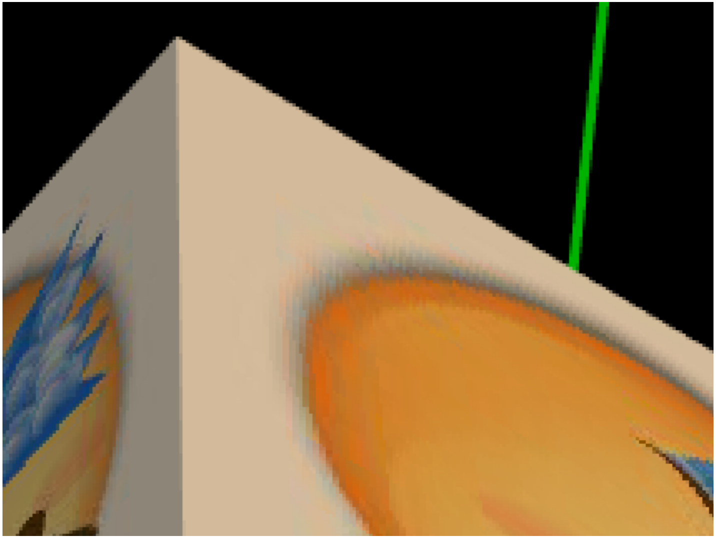
图形渲染中的几个原理

抗锯齿 (Anti-aliasing)



图形渲染中的几个原理

抗锯齿 (Anti-aliasing)



图形渲染中的几个原理

抗锯齿 (Anti-aliasing)

1. `glEnable(GL_POINT_SMOOTH);` 开启点的平滑;
2. `glEnable(GL_LINE_SMOOTH);` 开启直线的平滑;
3. `glEnable(GL_POLYGON_SMOOTH);` 开启多边形的平滑;
4. `glHint(GL_POINT_SMOOTH_HINT, GL_NICEST);` 设定平滑提示;
5. `glEnable(GL_MULTISAMPLE);` 开启多重采样抗锯齿.

1. 工程中的 `.c` 与 `.cpp` 文件, `.h` 与 `.hpp` 文件;
2. 各种各样的库;
3. 库中的头文件与链接库文件;
4. 动态链接库与静态链接库;
5. 编译器与编辑器, 集成开发环境 (IDE);
6. 命令行编译.

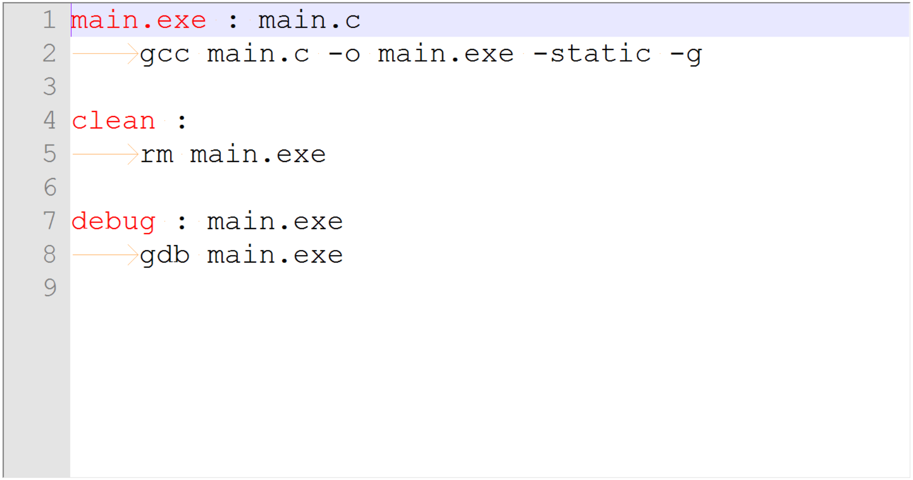
工程配置

Makefile 与 make

1. make 的基本作用;
2. Makefile 的简单语法;
3. make 命令.

工程配置

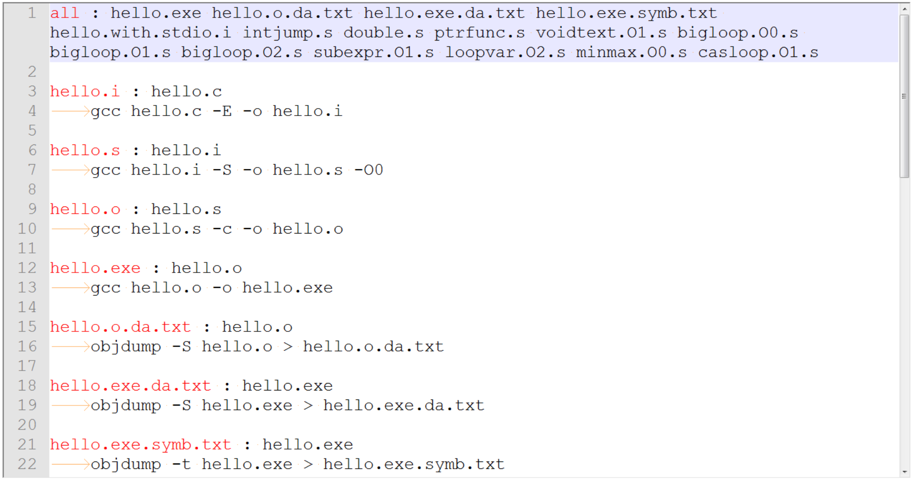
Makefile 与 make



```
main.exe : main.c
	gcc main.c -o main.exe -static -g

clean :
	rm main.exe

debug : main.exe
	gdb main.exe

```

工程配置

Makefile 与 make



```make
all : hello.exe hello.o.da.txt hello.exe.da.txt hello.exe.symb.txt hello.with.stdio.i intjump.s double.s ptrfunc.s voidtext.01.s bigloop.00.s bigloop.01.s bigloop.02.s subexpr.01.s loopvar.02.s minmax.00.s casloop.01.s

hello.i : hello.c
	gcc hello.c -E -o hello.i

hello.s : hello.i
	gcc hello.i -S -o hello.s -O0

hello.o : hello.s
	gcc hello.s -c -o hello.o

hello.exe : hello.o
	gcc hello.o -o hello.exe

hello.o.da.txt : hello.o
	objdump -S hello.o > hello.o.da.txt

hello.exe.da.txt : hello.exe
	objdump -S hello.exe > hello.exe.da.txt

hello.exe.symb.txt : hello.exe
	objdump -t hello.exe > hello.exe.symb.txt
```

工程配置

Makefile 与 make

1. make 只以时间为参照依据.

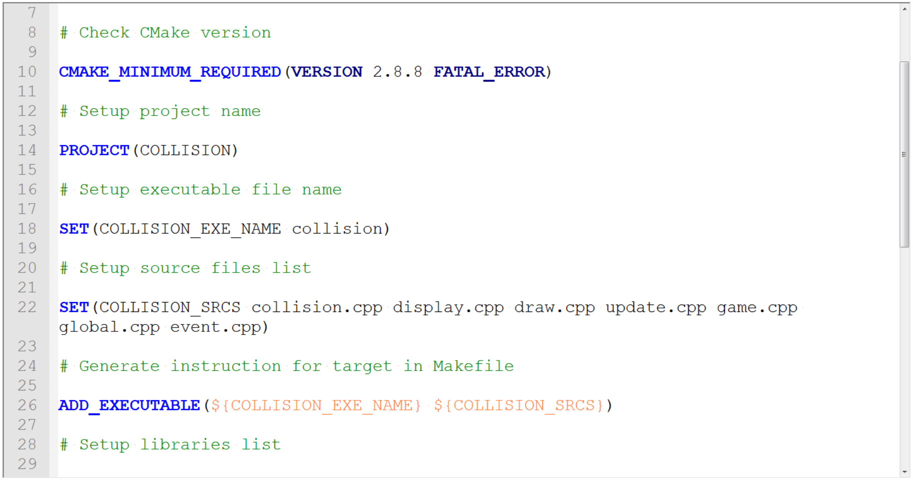
工程配置

CMake 与 CMakeLists.txt

1. CMake 的跨平台意义;
2. CMake 的基本语法;
3. cmake 命令.

工程配置

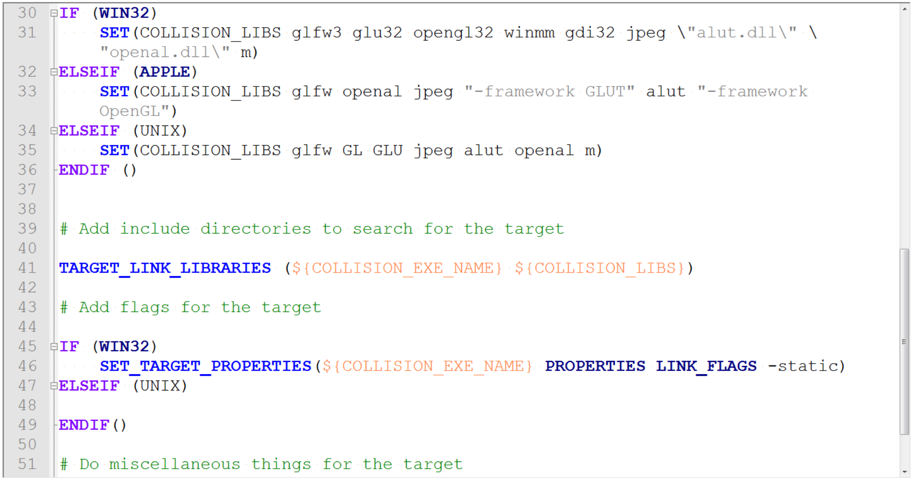
CMake 与 CMakeLists.txt



```

# Check CMake version

CMAKE_MINIMUM_REQUIRED(VERSION 2.8.8 FATAL_ERROR)

# Setup project name

PROJECT(COLLISION)

# Setup executable file name

SET(COLLISION_EXE_NAME collision)

# Setup source files list

SET(COLLISION_SRCS collision.cpp display.cpp draw.cpp update.cpp game.cpp
global.cpp event.cpp)

# Generate instruction for target in Makefile

ADD_EXECUTABLE(${COLLISION_EXE_NAME} ${COLLISION_SRCS})

# Setup libraries list

```

工程配置

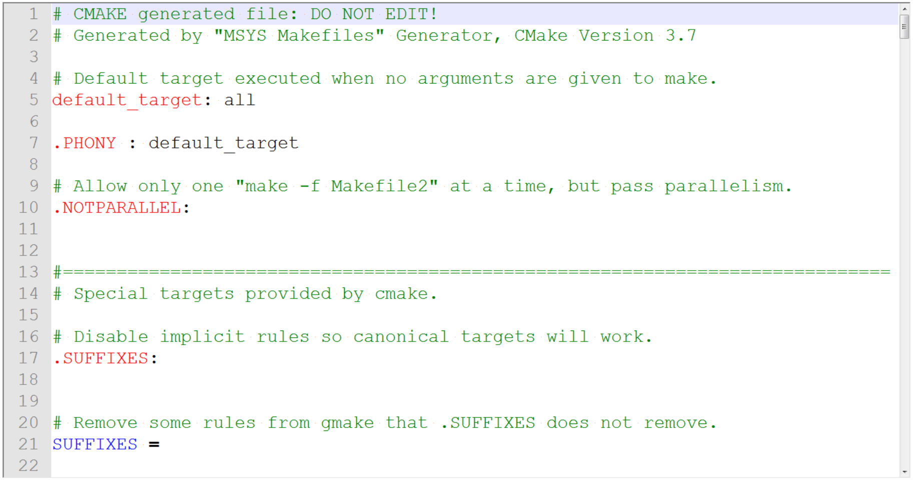
CMake 与 CMakeLists.txt



```
IF (WIN32)
    SET(COLLISION_LIBS glfw3 glu32 opengl32 winmm gdi32 jpeg \"alut.dll\" \"openal.dll\" m)
ELSEIF (APPLE)
    SET(COLLISION_LIBS glfw openal jpeg "-framework GLUT" alut "-framework OpenGL")
ELSEIF (UNIX)
    SET(COLLISION_LIBS glfw GL GLU jpeg alut openal m)
ENDIF ()


# Add include directories to search for the target

TARGET_LINK_LIBRARIES (${COLLISION_EXE_NAME} ${COLLISION_LIBS})

# Add flags for the target

IF (WIN32)
    SET_TARGET_PROPERTIES(${COLLISION_EXE_NAME} PROPERTIES LINK_FLAGS -static)
ELSEIF (UNIX)

ENDIF()

# Do miscellaneous things for the target
```

工程配置

CMake 与 CMakeLists.txt



```
# CMAKE generated file: DO NOT EDIT!
# Generated by "MSYS Makefiles" Generator, CMake Version 3.7

# Default target executed when no arguments are given to make.
default_target: all

.PHONY : default_target

# Allow only one "make -f Makefile2" at a time, but pass parallelism.
.NOTPARALLEL:


#=============================================================================
# Special targets provided by cmake.

# Disable implicit rules so canonical targets will work.
.SUFFIXES:


# Remove some rules from gmake that .SUFFIXES does not remove.
SUFFIXES =

```

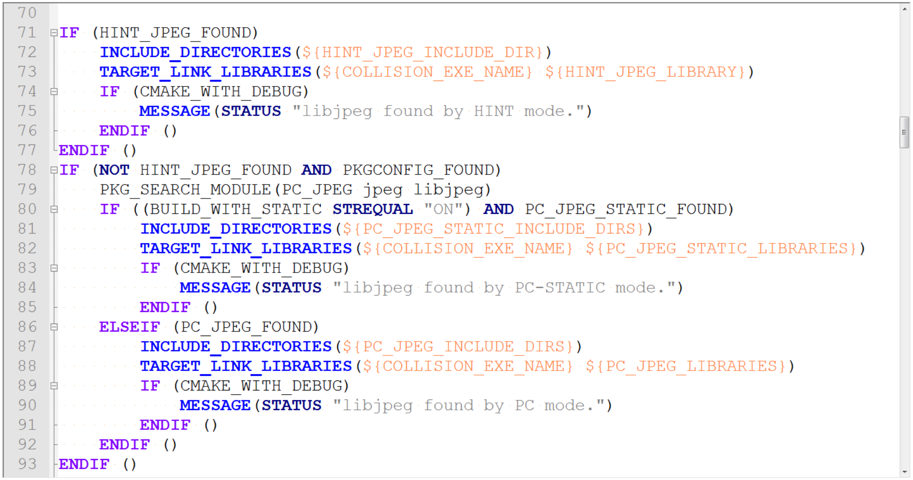
工程配置

CMake 与 CMakeLists.txt

1. cmake 的跨平台使用;
2. make install, glfw, gmp 等包的安装;
3. FIND_PACKAGE 命令带来了极大的便利;
4. pkg-config 工具;
5. make 彩色的编译界面.

工程配置

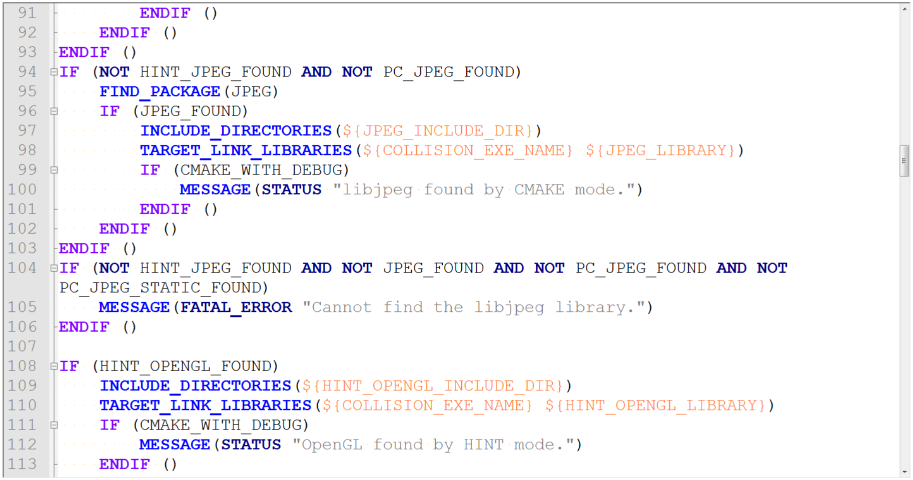
CMake 与 CMakeLists.txt



```
IF (HINT_JPEG_FOUND)
    INCLUDE_DIRECTORIES(${HINT_JPEG_INCLUDE_DIR})
    TARGET_LINK_LIBRARIES(${COLLISION_EXE_NAME} ${HINT_JPEG_LIBRARY})
    IF (CMAKE_WITH_DEBUG)
        MESSAGE(STATUS "libjpeg found by HINT mode.")
    ENDIF ()
ENDIF ()
IF (NOT HINT_JPEG_FOUND AND PKGCONFIG_FOUND)
    PKG_SEARCH_MODULE(PC_JPEG jpeg libjpeg)
    IF ((BUILD_WITH_STATIC STREQUAL "ON") AND PC_JPEG_STATIC_FOUND)
        INCLUDE_DIRECTORIES(${PC_JPEG_STATIC_INCLUDE_DIRS})
        TARGET_LINK_LIBRARIES(${COLLISION_EXE_NAME} ${PC_JPEG_STATIC_LIBRARIES})
        IF (CMAKE_WITH_DEBUG)
            MESSAGE(STATUS "libjpeg found by PC-STATIC mode.")
        ENDIF ()
    ELSEIF (PC_JPEG_FOUND)
        INCLUDE_DIRECTORIES(${PC_JPEG_INCLUDE_DIRS})
        TARGET_LINK_LIBRARIES(${COLLISION_EXE_NAME} ${PC_JPEG_LIBRARIES})
        IF (CMAKE_WITH_DEBUG)
            MESSAGE(STATUS "libjpeg found by PC mode.")
        ENDIF ()
    ENDIF ()
ENDIF ()
```

工程配置

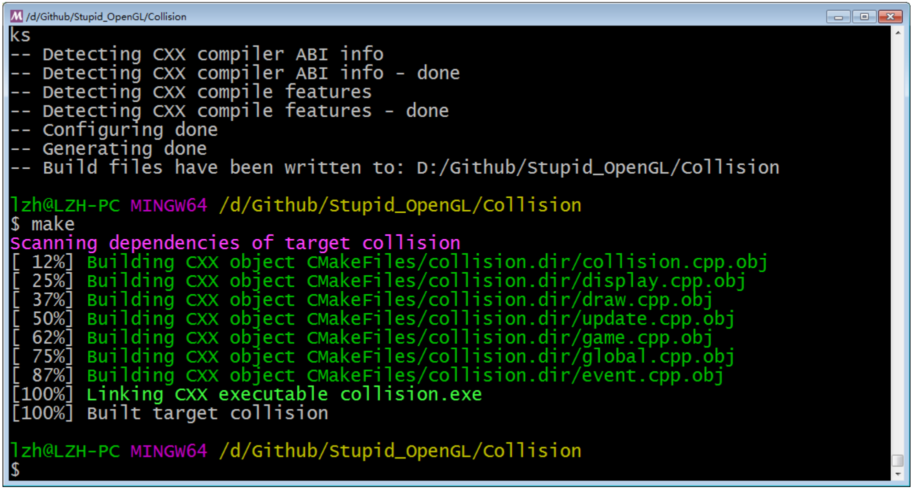
CMake 与 CMakeLists.txt



```
            ENDIF ()
        ENDIF ()
ENDIF ()
IF (NOT HINT_JPEG_FOUND AND NOT PC_JPEG_FOUND)
    FIND_PACKAGE(JPEG)
    IF (JPEG_FOUND)
        INCLUDE_DIRECTORIES(${JPEG_INCLUDE_DIR})
        TARGET_LINK_LIBRARIES(${COLLISION_EXE_NAME} ${JPEG_LIBRARY})
        IF (CMAKE_WITH_DEBUG)
            MESSAGE(STATUS "libjpeg found by CMAKE mode.")
        ENDIF ()
    ENDIF ()
ENDIF ()
IF (NOT HINT_JPEG_FOUND AND NOT JPEG_FOUND AND NOT PC_JPEG_FOUND AND NOT
PC_JPEG_STATIC_FOUND)
    MESSAGE(FATAL_ERROR "Cannot find the libjpeg library.")
ENDIF ()

IF (HINT_OPENGL_FOUND)
    INCLUDE_DIRECTORIES(${HINT_OPENGL_INCLUDE_DIR})
    TARGET_LINK_LIBRARIES(${COLLISION_EXE_NAME} ${HINT_OPENGL_LIBRARY})
    IF (CMAKE_WITH_DEBUG)
        MESSAGE(STATUS "OpenGL found by HINT mode.")
    ENDIF ()
```

工程配置

CMake 与 CMakeLists.txt



```
ks
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Configuring done
-- Generating done
-- Build files have been written to: D:/Github/Stupid_OpenGL/Collision

lzh@LZH-PC MINGW64 /d/Github/Stupid_OpenGL/Collision
$ make
Scanning dependencies of target collision
[ 12%] Building CXX object CMakeFiles/collision.dir/collision.cpp.obj
[ 25%] Building CXX object CMakeFiles/collision.dir/display.cpp.obj
[ 37%] Building CXX object CMakeFiles/collision.dir/draw.cpp.obj
[ 50%] Building CXX object CMakeFiles/collision.dir/update.cpp.obj
[ 62%] Building CXX object CMakeFiles/collision.dir/game.cpp.obj
[ 75%] Building CXX object CMakeFiles/collision.dir/global.cpp.obj
[ 87%] Building CXX object CMakeFiles/collision.dir/event.cpp.obj
[100%] Linking CXX executable collision.exe
[100%] Built target collision

lzh@LZH-PC MINGW64 /d/Github/Stupid_OpenGL/Collision
$
```