KOLKOIKRZYZYK PROGRAM REQUIREMENTS AND DOCUMENTATION

by Michał Listkowski and Radosław Sochalski

Spis treści

KolkoIKrzyzyk PROGRAM REQUIREMENTS

Primary actors:

- Human

Preconditions:

- One has PC with Windows / Linux Debian-like OS

Basic flow of events:

1. User starts the program and game board is automatically displayed.
2. User plays vs other user (changing sidy – by – side)

(human vs. human):

3. The program allows both human players to alternately select their moves by clicking demanded square (if it's not already taken by opposite player)
4. Human player 1 has O mark. The X mark goes to human player 2
5. Human player 1 always start firsts.

General gameplay:

6. Game must finish with win of a player. Opposite player will loose if player (1|2) wins.

Alternative flows:

1a. Option to let computer play with computer will be added later ☺

2a. Option 3 = exit and close program

3a. Numbers of tiles are displayed only at start of the game.

3a1. Numbers are displayed every move (taken tiles are marked with Xs or Os)

4a. If the tile is already taken, program will display a message and ask user to chose again.

5a. AI will be implemented as random algorithm chechking whether pool is taken or not and placing a mark on random basis

6a. Game will end with a draw.

Compiling and „make” / „cmake” structure

1. **Make**
   In main folder, there is a makefile: „makefile”
   Possible commands:
   - Make = creating binaries of game executable „KolkoIKrzyzyk“ and „runTest“ for gtest
   - Make all = same as above
   - Make clean – cleaning all created files

Makefile structure:

all: KolkoIKrzyzyk runTest

KolkoIKrzyzyk: main.o MyGame.o

g++ main.o MyGame.o -o KolkoIKrzyzyk -lsfml-graphics -lsfml-window -lsfml-system

runTest: main_test.o MyGame.o

g++ main_test.o MyGame.o -o runTest -L/usr/local/include/gtest -lgtest -lgtest_main -lsfml-graphics -lsfml-window -lsfml-system

main_test.o: main_test.cpp

g++ -std=c++11 -c main_test.cpp

main.o: main.cpp

g++ -std=c++11 -c main.cpp

MyGame.o: MyGame.cpp MyGame.h

g++ -std=c++11 -c MyGame.cpp MyGame.h

clean:

rm -rf *.o *.gch KolkoIKrzyzyk runTest

2. **CMake:**
   - as standard create a directory to make a build
   - cmake will only compile main game executable!

CMakeLists.txt structure:

```
CMake_minimum_required (VERSION 2.6)

project(KolkoIKrzyzyk)

#add_executable(KolkoIKrzyzyk main.cpp MyGame.cpp MyGame.h)
add_definitions(-std=c++11)
include_directories(${PROJECT_BINARY_DIR})
```

```
# Enable debug symbols by default
# must be done before project() statement
if(NOT CMAKE_BUILD_TYPE)
  set(CMAKE_BUILD_TYPE Debug CACHE STRING "Choose the type of build (Debug
or Release)" FORCE)
endif()

# Define sources and executable
set(EXECUTABLE_NAME "TheGame")
add_executable(${EXECUTABLE_NAME} main.cpp MyGame.cpp)

# Detect and add SFML
set(CMAKE_MODULE_PATH "${PROJECT_SOURCE_DIR}/cmake_modules"
${CMAKE_MODULE_PATH})
#Find any version 2.X of SFML
#See the FindSFML.cmake file for additional details and instructions
#set(SFML_STATIC_LIBRARIES TRUE)
find_package(SFML REQUIRED graphics window system)
if(SFML_FOUND)
  include_directories(${SFML_INCLUDE_DIR})
  target_link_libraries(${EXECUTABLE_NAME} ${SFML_LIBRARIES}
${SFML_DEPENDENCIES})
endif()

#FONT
file(COPY ${PROJECT_SOURCE_DIR}/Xeron.ttf DESTINATION
${CMAKE_CURRENT_BINARY_DIR})

# Install target
install(TARGETS ${EXECUTABLE_NAME} DESTINATION bin)
```

COMPILING PREREQUISITES:

1. SFML v2.4.2 downloaded and installed (cmake + make + make install)
   For Debian-like systems:
   $> sudo apt-get install libsfml-dev
2. Googletest downloaded and installed (cmake + make + make install)
   For github:
   git clone https://github.com/google/googletest.git
   > descent into folder
   > cmake + make + make install
3. Windows not supported, however it works with Windows and Visual Studio 13+ (SFML libraries Path must be properly linked and in main.cpp you will have to change #include „SFML/Graphics.hpp“ to „SFML\Graphics.hpp“)

GTEST automatic Tests output:

1. Tests are initialized within main_test.cpp file.
2. Output of gtest in ./runTest is placed in https://github.com/list3k/KiK_linux/gtest_output.txt
3. Direct output:

```
[==========] Running 6 tests from 2 test cases.
[----------] Global test environment set-up.
[----------] 3 tests from handleClick
[ RUN      ] handleClick.clicked_top_left_0_0
[       OK ] handleClick.clicked_top_left_0_0 (54 ms)
[ RUN      ] handleClick.clicked_middle_1_1
[       OK ] handleClick.clicked_middle_1_1 (27 ms)
[ RUN      ] handleClick.not_clicked
[       OK ] handleClick.not_clicked (21 ms)
[----------] 3 tests from handleClick (102 ms total)

[----------] 3 tests from isWinner
[ RUN      ] isWinner.a_WIN
[       OK ] isWinner.a_WIN (26 ms)
[ RUN      ] isWinner.not_a_WIN
[       OK ] isWinner.not_a_WIN (18 ms)
[ RUN      ] isWinner.a_TIE
[       OK ] isWinner.a_TIE (17 ms)
[----------] 3 tests from isWinner (61 ms total)

[----------] Global test environment tear-down
[==========] 6 tests from 2 test cases ran. (163 ms total)
[  PASSED  ] 6 tests.
```

Above tests are checking 2 bool functions as all other functions are only drawing objects.