SSI LINUX &ANDROID PORT LOG

SIMON FLUTURA

CONTENTS

1 PORTED PLUGINS AND NOTES

1.1 ssiframe

1.2 ssievent

1.3 ssiioput

1.4 ssimouse

This plugin uses libxinput2 on linux, thus depends on xorg. Not ported to Android!

1.5 ssiaudio

This plugin depends on portaudio on linux and
OpenSE on android.

1.6 ssimodel

1.7 ssisignal

1.8 ssiemovoice

1.9 ssiopensmile

not tested

1.10 ssiandroidsensors

Plugin only avaliable on Android. Enables messages via logcat.

```
ssimsg=new ssi::AndroidMessage();
```

1.11 ssiandroidjavasensors

Android only. Sends java events to ssi. E.g bluetooth or battery.

1.12 ssiwebsockets

1.13 ssivectorfusion

1.14 ssiffmpeg

not tested

2 SSJ INTEGRATION

for ssj integration a ssj sensor plugin is added to ssi; int is wrapped on ssj side via sensorProviderSSI, that replaces ssj core with ssi.

other posibilities include running both frameworks in parallel wher an ssj consumer is paired to the ssi ssj sensor.

following JNI snipplets might be helpful:

3 APK BUILD SUPPORT

see android doc for build instructions. SSI uses android.apk.cmake to create an APK android package. ANT is used; todo switch to maven?

At the moment the APK is build around `android_xmlpipe` found in `plugins/androidSensors/tools/x` The java code found in `docs/apk` is used to extract native libraries then the android main activity in `android_xmlpipe` is started. After extracting the pipeline using native assetmanager, xmlpipe is started as usual.

4 SSI CMAKE BUILDSYSTEM

Old Visual Studio Buildsystem is untouched, but Visual Studio Projects can be generated from Cmake.

Cmake generates CodeBlocks make projects, linux make projects.

4.1 Structure

Each project has to live in its own subdirectory for better cmake integration. Each subproject has its own CMakeLists.txt file. Every library or executeable is a subproject of its own to keep things simple.

A directory containing CMake-projects has to contain a CMakeLists.txt mentioning subpojects.

Variables for SSI install path, `plattform/compiler` detection etc can be found in the `trunk/CMakeLists.txt`

4.2 how to add plugin source

Add subdirectory to `trunk/plugins/CMakeLists.txt` with `add_subdirectory(dirname)`.

Add a CMakeLists.txt:

```
# add project
project(ssimouse)

#add subdirectory for text
add_subdirectory(test)
#add tests dependencies
add_dependencies(ssimouse_test ssi ssimouse)


#add include directories
include_directories (
  include
  ../../core/include
  ../../core/include/ioput/socket
  ../../core/include/ioput

  ../event/include

  ../frame/include

  ../graphic/include

  ../ioput/include
  ../ioput/include/ioput/socket
  ../ioput/include/ioput
  ../
)
```

```
#set source files
set(COMMON_SRC_FILES

 source/ExportMain.cpp
 source/Mouse.cpp
 source/CursorMover.cpp

)

#find librarys

IF(MINGW)
  find_library (MINGW_WSOCKET "wsock32" HINTS ${MINGWLIB} )
  find_library (MINGW_WMM "winmm" HINTS ${MINGWLIB})
  find_library (MINGW_WSOCKET2 "ws2_32" HINTS ${MINGWLIB} )
  find_library (MINGW_PTHREAD "pthread" HINTS ${MINGWLIB} )
  #set compiler flags for c++11 threading and debug
  #todo create gcc/make debug target?
    set( CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -std=c++11 -ggdb" )

  ELSEIF(UNIX)
  SET(MINGW_WSOCKET "")
  SET(MINGW_WSOCKET2 "")
  SET(MINGW_WMM "")

  set( CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -std=c++11 -ggdb" )

  find_library (MINGW_PTHREAD "pthread" HINTS "/usr/lib" )
  # x11 dependencies
  find_library (X11 "X11" HINTS "/usr/X11R6/lib")
  find_library (Xi "Xi" HINTS "/usr/X11R6/lib")

ELSE(MINGW) # visual studio has its own ways for winsockets/threading
    eg #pragma
  SET(MINGW_WSOCKET "")
  SET(MINGW_WSOCKET2 "")
  SET(MINGW_WMM "")
   SET(MINGW_PTHREAD "")
ENDIF(MINGW)


set(SRC_FILES ${COMMON_SRC_FILES} )

# add main target
add_library(ssimouse SHARED ${SRC_FILES})

# link against internal and external librarys
IF(UNIX)
target_link_libraries(ssimouse ssi ${MINGW_WSOCKET} ${MINGW_WSOCKET2}
    ${MINGW_WMM} ${MINGW_PTHREAD})
ELSE(UNIX)
target_link_libraries(ssimouse ssi ${MINGW_WSOCKET} ${MINGW_WSOCKET2}
    ${MINGW_WMM} ${MINGW_PTHREAD} ${X11} ${Xi})
ENDIF(UNIX)

#add_executable(myapp main.c)

#rename targets if debug
set_target_properties(ssimouse PROPERTIES DEBUG_POSTFIX "d" PREFIX ""
    )

#install target to ssi install path set in base directory
install(TARGETS ssimouse DESTINATION ${SSI_INSTALL}/${SSI_PLATFORM}/$
    {SSI_COMPILER})
```

Listing 1: cmake ssi plugin

4.3 how to add a plugin library

The library has to be added to the ssi directory tree in trunk/libs/shared/libname. Binarys have to be added into according plattform and compiler subdirs. The library should contain a macro for easier use in projects, if it contains multiple files.

Thereafter the library can be added to the project:

```
# add project
project(ssishore)

#add test
add_subdirectory(test)
add_dependencies(ssishore_test ssi ssishore)

# add include dirs
include_directories (
  include
  ../../core/include
  ../../libs/shared/opencv/include/
  ../../plugins
)

# set source files
set(COMMON_SRC_FILES

source/ExportMain.cpp

)


set(SRC_FILES ${COMMON_SRC_FILES} )


# set libraries libs path
get_filename_component(OPENCV_PATH ../../libs/shared/opencv/libs/${
    SSI_PLATFORM}/${SSI_COMPILER} ABSOLUTE)
# set libraries bin path
get_filename_component(OPENCV_PATH_SHARED ../../libs/shared/opencv/
    bin/${SSI_PLATFORM}/${SSI_COMPILER} ABSOLUTE)


# use macro to find precompiled opencv libs
include(../../libs/shared/opencv/opencv_paths.cmake)

opencvPaths(${OPENCV_PATH} ${OPENCV_PATH_SHARED} OPENCV_LIB_DEBUG
    OPENCV_LIB OPENCV_SHARED_DEBUG OPENCV_SHARED)


# find single file static library for linking
find_library(
      SHORE_LIB

      NAMES

        shore140.lib

      HINTS
        ../../libs/shared/shore/libs/${SSI_PLATFORM}/${SSI_COMPILER}

      )

# find single file dynamic library for copiing
find_file(
      SHORE_SHARED

```

```
        NAMES

          shore140.dll

        HINTS
          ../../libs/shared/shore/bin/${SSI_PLATFORM}/${SSI_COMPILER}

        )

# add main target

add_library(ssishore SHARED ${SRC_FILES})


# link for debug
target_link_libraries(ssishore debug ssi
                            ${OPENCV_LIB_DEBUG} ${SHORE_LIB}
                            )

# link for release
target_link_libraries(ssishore optimized ssi
                            ${OPENCV_LIB} ${SHORE_LIB}
                            )
#add_executable(myapp main.c)

#rename targets if debug
set_target_properties(ssishore PROPERTIES DEBUG_POSTFIX "d")

#install target to ssi install path set in base directory
install(TARGETS ssishore DESTINATION ${SSI_INSTALL}/${SSI_PLATFORM}/${SSI_COMPILER})

#install dll files (copy)
install(FILES ${SHORE_SHARED} ${OPENCV_SHARED} ${OPENCV_SHARED_DEBUG}
      DESTINATION ${SSI_INSTALL}/${SSI_PLATFORM}/${SSI_COMPILER})
```

Listing 2: cmake ssi plugins library

Here follows the macro for adding multiple lib files:

```
function( opencvPaths OPENCV_PATH OPENCV_PATH_SHARED OPENCV_LIB_DEBUG
      OPENCV_LIB OPENCV_SHARED_DEBUG OPENCV_SHARED)

  # add static debug libs
  if(WIN32)
    set(${OPENCV_LIB_DEBUG}

                ${OPENCV_PATH}/opencv_core2410d.lib
                ${OPENCV_PATH}/opencv_features2d2410d.lib

            PARENT_SCOPE)
  endif(WIN32)

  # add static release libs
  if(WIN32)
    set(${OPENCV_LIB}

                ${OPENCV_PATH}/opencv_core2410.lib
                ${OPENCV_PATH}/opencv_features2d2410.lib

                PARENT_SCOPE)
  endif(WIN32)

  # add debug dlls
  if(WIN32)
    set(${OPENCV_SHARED_DEBUG}
```

```
28                  ${OPENCV_PATH_SHARED}/opencv_core2410d.dll
                    ${OPENCV_PATH_SHARED}/opencv_features2d2410d.dll
30
32                  PARENT_SCOPE)
  endif(WIN32)
34
  # add release dlls
36 if(WIN32)
    set(${OPENCV_SHARED}
38
                    ${OPENCV_PATH_SHARED}/opencv_core2410.dll
40                  ${OPENCV_PATH_SHARED}/opencv_features2d2410.dll

42                  PARENT_SCOPE)
  endif(WIN32)
44

46 endfunction(opencvPaths)
```

Listing 3: cmake library macro

4.4 how to add a source file to Core

Simply add the file to the `COMMON_SRC_FILES` list in trunk/core/CMakeList.txt. If the file contains code that is only used on one plattform, add it to `P_SRC_FILES` in the according case.

```
2 set(COMMON_SRC_FILES

4 source/buffer/Buffer.cpp

6 )
  IF(WIN32)
8 set(P_SRC_FILES
   source/ioput/socket/ip/win32/NetworkingUtils.cpp
10  )
  ELSE(WIN32)
12 set(P_SRC_FILES
   source/ioput/socket/ip/posix/NetworkingUtils.cpp
14

16  )
  ENDIF(WIN32)
```

Listing 4: add source file to core

4.5 how to add a test or tool

Tests are a subproject of their own. They need to be added to the parent directory.

```
1 # add project
  project(ssiframe_test)
3
  # add incluede dirs
5 include_directories (
    .
7   ../../../core/include
    ../../../core/include/ioput/socket
9   ../../../core/include/ioput

11  ../../../plugins/
```

```
)

# add source files
set(COMMON_SRC_FILES

 Main_.cpp
)


set(SRC_FILES ${COMMON_SRC_FILES} )

#find librarys

IF(MINGW)
   find_library (MINGW_WSOCKET "wsock32" HINTS ${MINGWLIB} )
   find_library (MINGW_WMM "winmm" HINTS ${MINGWLIB})
   find_library (MINGW_WSOCKET2 "ws2_32" HINTS ${MINGWLIB} )
   find_library (MINGW_PTHREAD "pthread" HINTS ${MINGWLIB} )
   #set compiler flags for c++11 and debug
   #todo debug target on gcc
     set( CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -std=c++11 -ggdb" )

   ELSEIF(UNIX)
   SET(MINGW_WSOCKET "")
   SET(MINGW_WSOCKET2 "")
   SET(MINGW_WMM "")
   set( CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -std=c++11 -ggdb" )
   find_library (MINGW_PTHREAD "pthread" HINTS "/usr/lib" )
   find_library (X11 "X11" HINTS "/usr/X11R6/lib")
   find_library (Xi "Xi" HINTS "/usr/X11R6/lib")

ELSE(MINGW)
   SET(MINGW_WSOCKET "")
   SET(MINGW_WSOCKET2 "")
   SET(MINGW_WMM "")
    SET(MINGW_PTHREAD "")
ENDIF(MINGW)

# add projects main target (library or binary)
add_executable(ssiframe_test ${SRC_FILES})

# add linking to other projects or external librarys
target_link_libraries(ssiframe_test ssi ${MINGW_WSOCKET} ${MINGW_WMM}
      ${MINGW_WSOCKET2} ${MINGW_PTHREAD})


#rename targets if debug
set_target_properties(ssiframe_test PROPERTIES DEBUG_POSTFIX "d")

#install target to ssi install path set in base directory
install(TARGETS ssiframe_test DESTINATION ${SSI_INSTALL}/${
    SSI_PLATFORM}/${SSI_COMPILER})
```

Listing 5: cmake subproject

4.6 compiler parameters

for gcc c++11 threading

4.7 doxygen

`http://majewsky.wordpress.com/2010/08/14/tip-of-the-day-cmake-and-doxygen/`

4.8 tests

`http://mifrosu.blogspot.de/2013/02/cmake-and-google-test-framework.html`

5 THREADING

Cx11 on Android. Just depends on version of gcc compiler(4.8+).

5.1 c++11

std::thread contains conditional variables
problem: interruptable threads

5.2 windows

old ssi threading system is untouched

5.3 posix

are implemented via c++11 interruptable threads can be implemented via pthread-cancle and pthread-kill

6 TIMER

Timers need a plattform independent abstraction. Windows has its own timers, linux uses Posix standard, android features its own challanges as suspend disturbs the timers.
boost.timer?
`std::chrono::high resolution clock?`
`http://stackoverflow.com/questions/1487695/c-cross-platform-high-resolution-timer`
linux monolitic raw

7 SOCKETS

on linux udp sockets might only be able to send, when their data is recived due to icmp packages.

8 FILE TOOLS

file tools for selecting all files in subdir and especially file dialogs have to be ported.

9 NAMED PIPES

named pipes are not yet ported.

10 DECENTRALIZATION & SYNCRONIZATION?

already features for streaming ssi info ssi.
synchronisation using ms since 1970? `std::chrono::time_point::time_since_epoch`
needs 64 bit:
`uint32_t` ms overflow after 50 days `uint64_t` ms overflow after 584942417 years

11 GUI REWRITE

The plotting part needs to be rewritten.

11.1 sdl2

multiwindow since 2.0. has support on many plattforms, win, linux, android, osx.
no support for multithreaded rendering: use cairo for rendering, sdl2 to display
sdl windowmanager runs its own thread, all sdl calls should happen there.

11.2 windowmanagment linux

multi window and (multi)console handling

11.3 qt

big dependency.

11.4 html5 & websockets

12 BUILDING SSI FOR ANDROID

Note: this needs Android-NDK set up on your system. follow instructions from /docs/ssi-port-cmake/intro-android(-from-win)

13 CROSSCOMPILATION

1. crosscompile linux -> arm linux (android) see android and win-android documentation for an up to date tutorial.
2. linux -> win32/64 mingw
   crosscompilation via mxe: `cmake ../trunk/ -DCMAKE_TOOLCHAIN_FILE=~/mxe/usr/i686-w64-ming`
3. windows -> arm linux (android)?

14 BUILDSYSTEMS OVERVIEW

A main problem of making a project portable is choosing the right buildsystem. The SSI requirements would be integration into multiple IDEs as Visual Studio and Code::Blocks as well as crossplattform support with windows, android and Linux support and crosscompiling eg Linux to Android.

14.1 Cmake

biggest community. fullfills requirements. Module scripts necessary. Bad automatisation, needs to be adjusted per hand, hard to update cmake from visual studio.

14.2 Gyp

Googles Buildsystem might be an option

14.3 Waf

Written in python, adjustable. Fullfills all requirements but is not as widely used as cmake. Also suffers from bad integration into IDEs.

14.4 Jam and others

Projects such as Boost use jam for (cross plattform) building. Many different versions therefore fragmented community.

Same goes for several other buildsystems.

15 BUGS WORTH MENTIONING

15.1 Singleton reinitialization on Android

ssi factory does not get destroyed when backgroundservice is stopped. constructor is not called, structs have to be reinitialized manually!

15.2 Memory leak in Linux mouse-plugin

free memory allocated by x11 function XIQueryPointer() manually. `free(buttons.mask);`

15.3 Memory corruption in Linux SignalPainter

included wrong header. memory got allocated using new compiled symbol but old struct was expected.

15.4 Mutex violation with c++11 condition variables

c++11 condition variables use std::locks instead of mutexes. creating locks on the fly leads to opening the mutex in the wrong place.

use `condition_variable_any` instead.

15.5 UDP Broadcast not working

setEnableBroadcast $socket_optwasnotsetinsendbutonlyinsendto$

15.6 Memory corruption in Linux EventMonitor