Tutorial Practice Session

Step 1: OpenVX Basics

Material for this tutorial

- **Utils**
  - VirtualBox

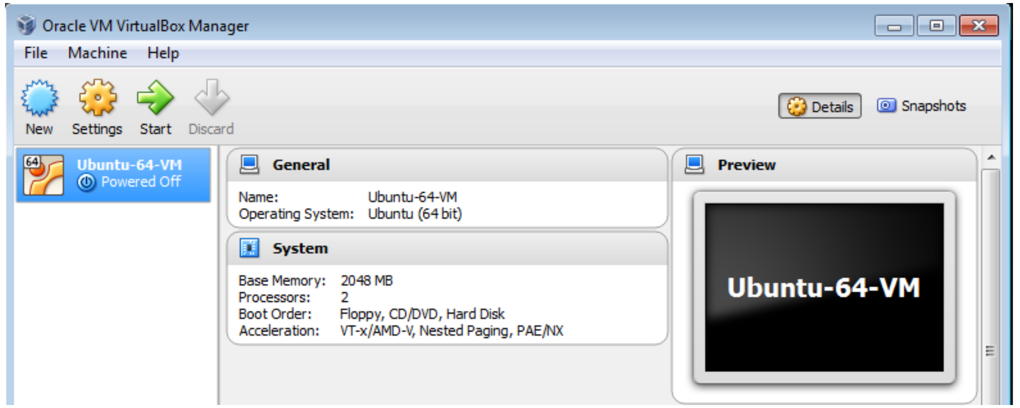
- **Tutorial VirtualBox image**
  - Khronos OpenVX ‘Sample’ implementation: functional OpenVX implementation
  - Open source OpenVX implementation
  - Source code for the different steps
  - Tutorial Videos
  - OpenVX 1.0.1 specification (pdf)
  - IDE: qtcreator

- **OpenVX spec in HTLM: www.khronos.org/registry/vx/specs/1.0.1/html**

Setup: Booting Ubuntu

- Uncompress Ubuntu-64-OpenVX.zip
- Install & run VirtualBox
- Add & run the VM image
  - Machine -> Add -> Ubuntu-64-OpenVX.vbox
  - Click 'Start'

- ***Some useful commands***
  - *Tab* *auto-complete a command*
  - *Ctrl-Alt-T* : *Open a terminal*
  - `ll` : *list files*
  - `cd <dir>` : *enter directory*
  - `cd ..` : *parent directory*
  - `gedit <file>` : *editor*
  - `evince <file>.pdf` : *pdf viewer*

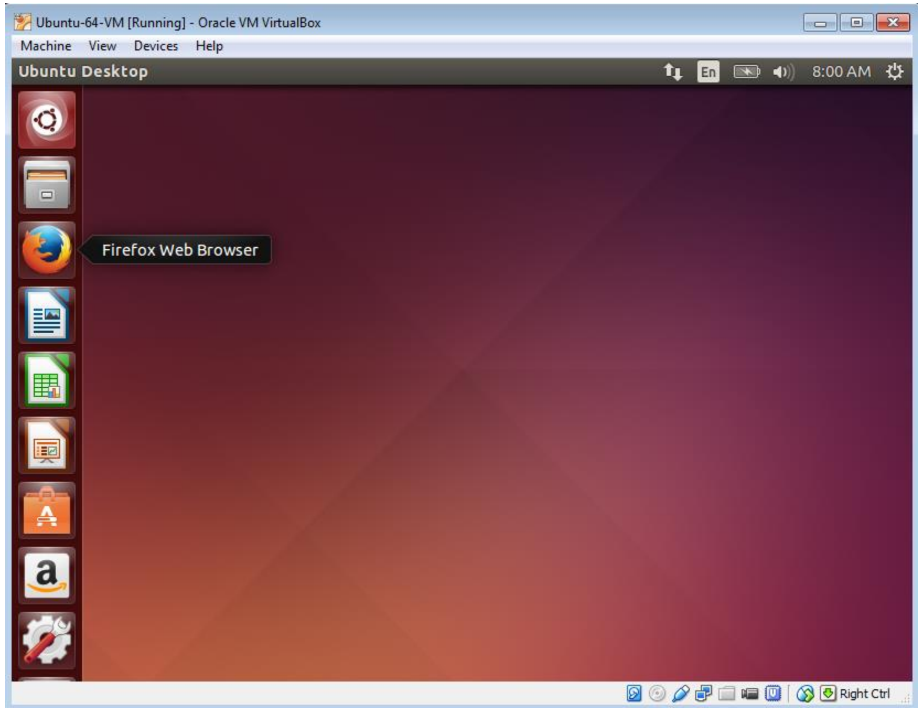


General

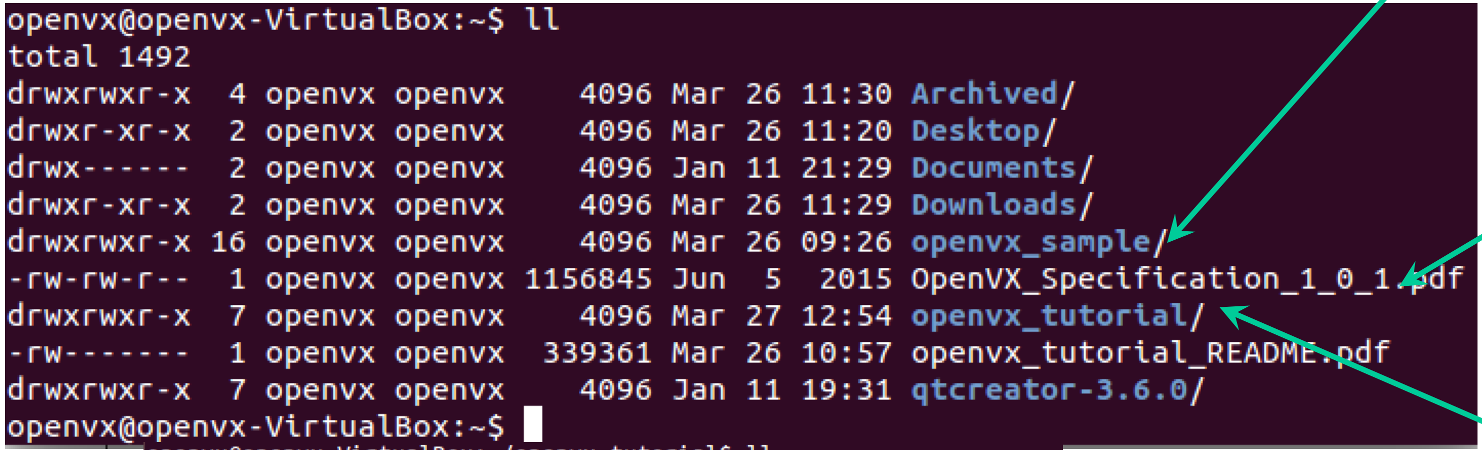
Name: Ubuntu-64-VM
Operating System: Ubuntu (64 bit)

System

Base Memory: 2048 MB
Processors: 2
Boot Order: Floppy, CD/DVD, Hard Disk
Acceleration: VT-x/AMD-V, Nested Paging, PAE/NX



Setup: Directory Structure

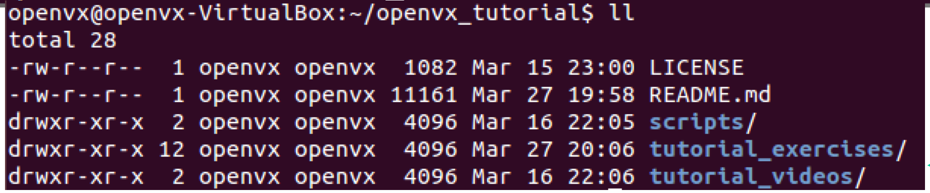


```
openvx@openvx-VirtualBox:~$ ll
total 1492
drwxrwxr-x  4 openvx openvx    4096 Mar 26 11:30 Archived/
drwxr-xr-x  2 openvx openvx    4096 Mar 26 11:20 Desktop/
drwx------  2 openvx openvx    4096 Jan 11 21:29 Documents/
drwxr-xr-x  2 openvx openvx    4096 Mar 26 11:29 Downloads/
drwxrwxr-x 16 openvx openvx    4096 Mar 26 09:26 openvx_sample/
-rw-rw-r--  1 openvx openvx 1156845 Jun  5  2015 OpenVX_Specification_1_0_1.pdf
drwxrwxr-x  7 openvx openvx    4096 Mar 27 12:54 openvx_tutorial/
-rw-------  1 openvx openvx  339361 Mar 26 10:57 openvx_tutorial_README.pdf
drwxrwxr-x  7 openvx openvx    4096 Jan 11 19:31 qtcreator-3.6.0/
openvx@openvx-VirtualBox:~$
```

Khronos Sample Implementation

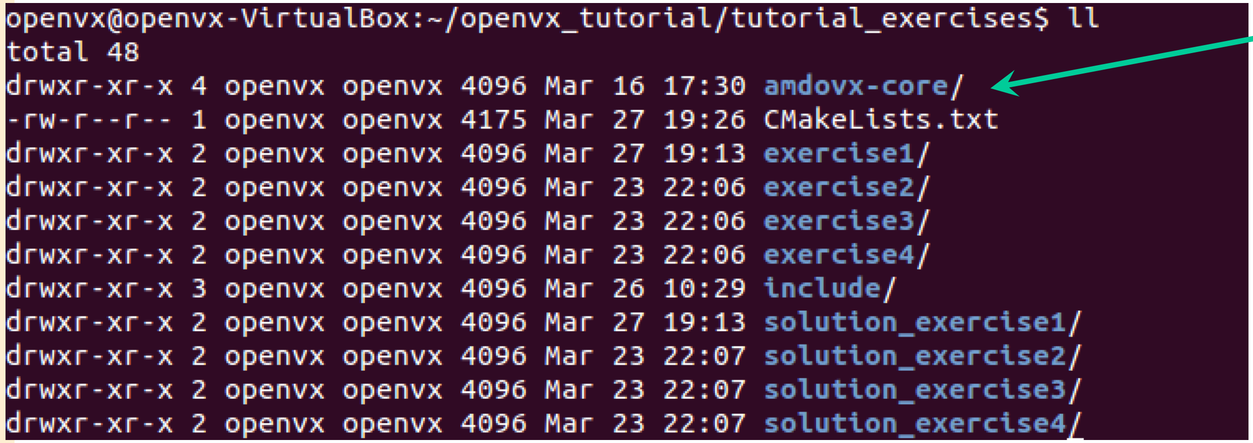
OpenVX specification

Tutorial source code



```
openvx@openvx-VirtualBox:~/openvx_tutorial$ ll
total 28
-rw-r--r--  1 openvx openvx  1082 Mar 15 23:00 LICENSE
-rw-r--r--  1 openvx openvx 11161 Mar 27 19:58 README.md
drwxr-xr-x  2 openvx openvx  4096 Mar 16 22:05 scripts/
drwxr-xr-x 12 openvx openvx  4096 Mar 27 20:06 tutorial_exercises/
drwxr-xr-x  2 openvx openvx  4096 Mar 16 22:06 tutorial_videos/
```

Tutorial video example



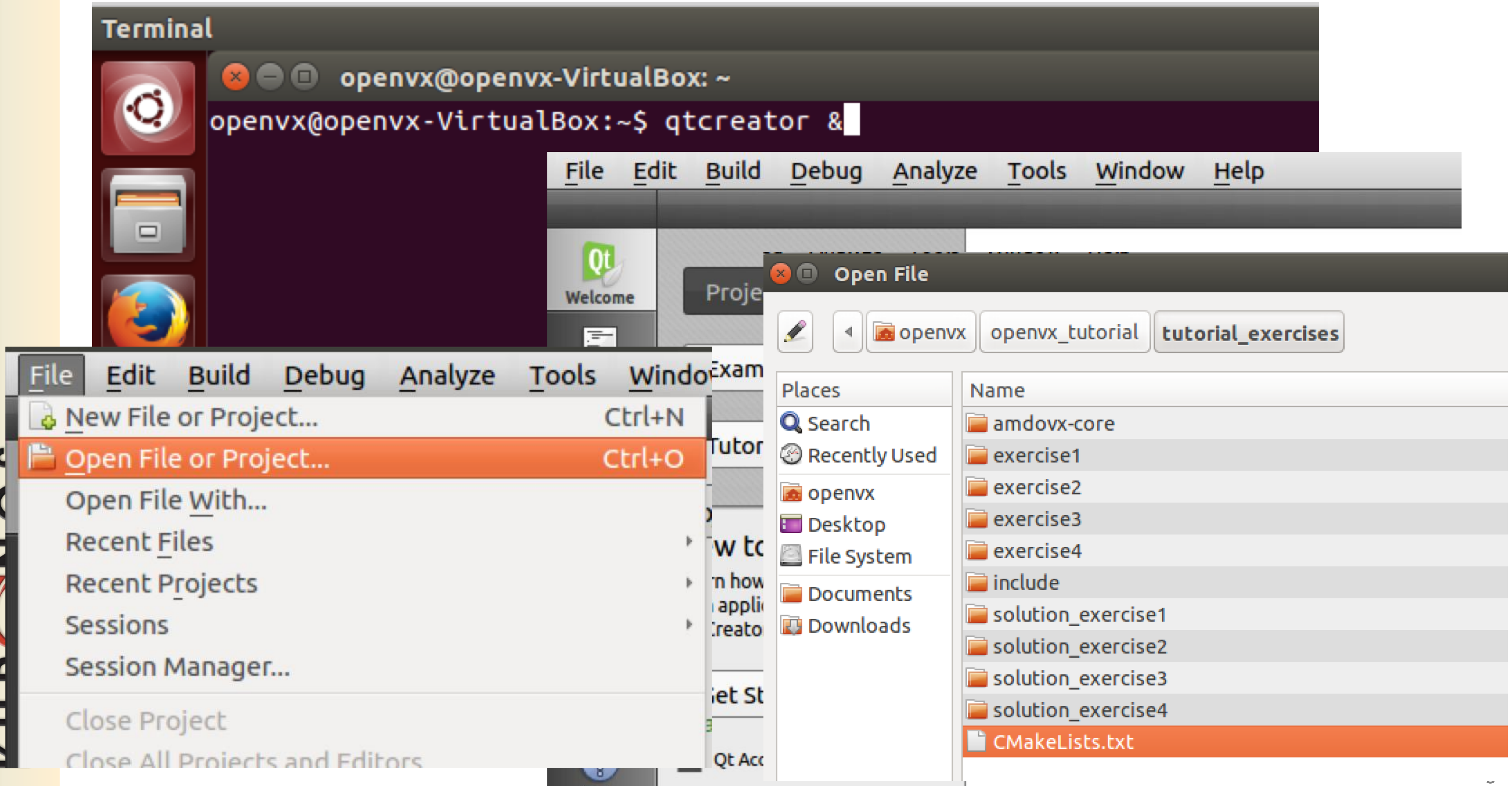
```
openvx@openvx-VirtualBox:~/openvx_tutorial/tutorial_exercises$ ll
total 48
drwxr-xr-x 4 openvx openvx 4096 Mar 16 17:30 amdovx-core/
-rw-r--r-- 1 openvx openvx 4175 Mar 27 19:26 CMakeLists.txt
drwxr-xr-x 2 openvx openvx 4096 Mar 27 19:13 exercise1/
drwxr-xr-x 2 openvx openvx 4096 Mar 23 22:06 exercise2/
drwxr-xr-x 2 openvx openvx 4096 Mar 23 22:06 exercise3/
drwxr-xr-x 2 openvx openvx 4096 Mar 23 22:06 exercise4/
drwxr-xr-x 3 openvx openvx 4096 Mar 26 10:29 include/
drwxr-xr-x 2 openvx openvx 4096 Mar 27 19:13 solution_exercise1/
drwxr-xr-x 2 openvx openvx 4096 Mar 23 22:07 solution_exercise2/
drwxr-xr-x 2 openvx openvx 4096 Mar 23 22:07 solution_exercise3/
drwxr-xr-x 2 openvx openvx 4096 Mar 23 22:07 solution_exercise4/
```

Open source Implementation

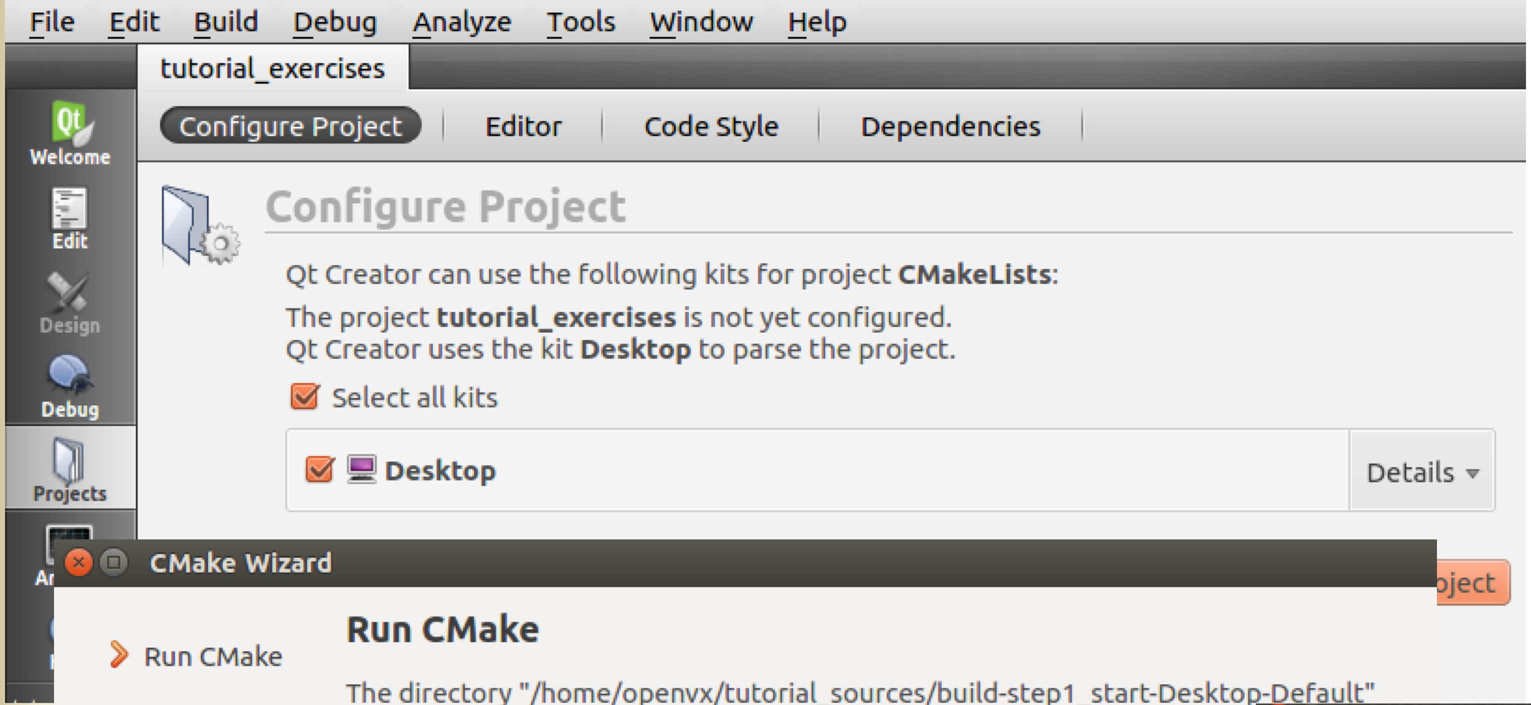
Source code for the tutorial

- `exercise<n>`	starting point
- solution_exercise`<n>`	full solution

Open a terminal with CTRL + ALT + T



```
openvx@openvx-VirtualBox:~$ qtcreator &
```



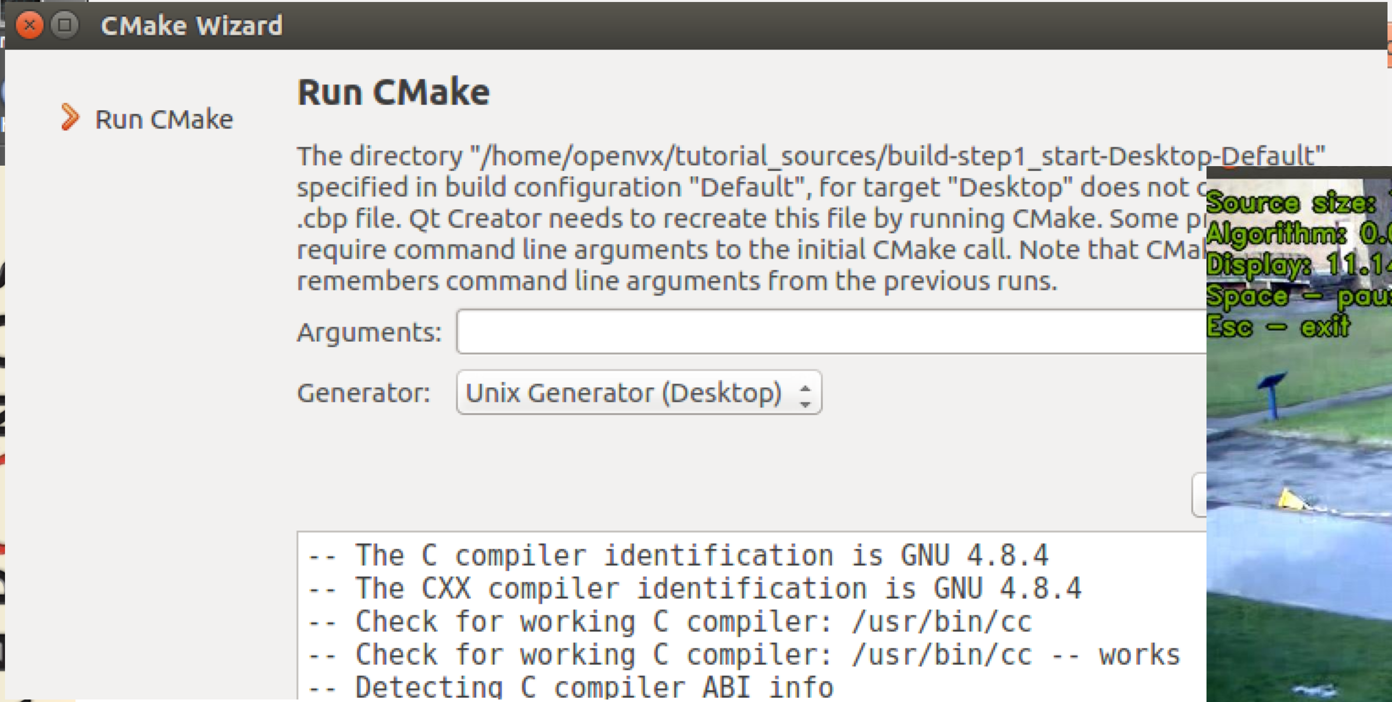
Configure Project

Qt Creator can use the following kits for project **CMakeLists**:

The project **tutorial_exercises** is not yet configured.
Qt Creator uses the kit **Desktop** to parse the project.

☑ Select all kits

☑ **Desktop** Details



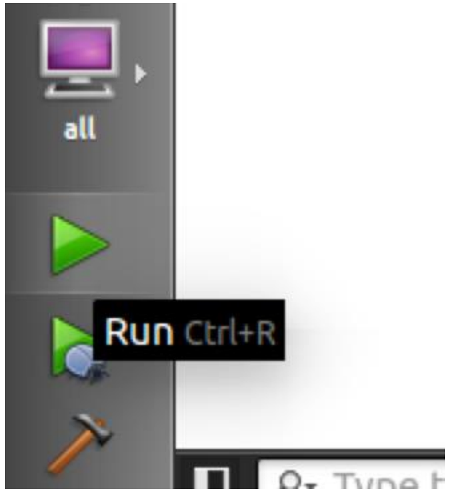
Run CMake

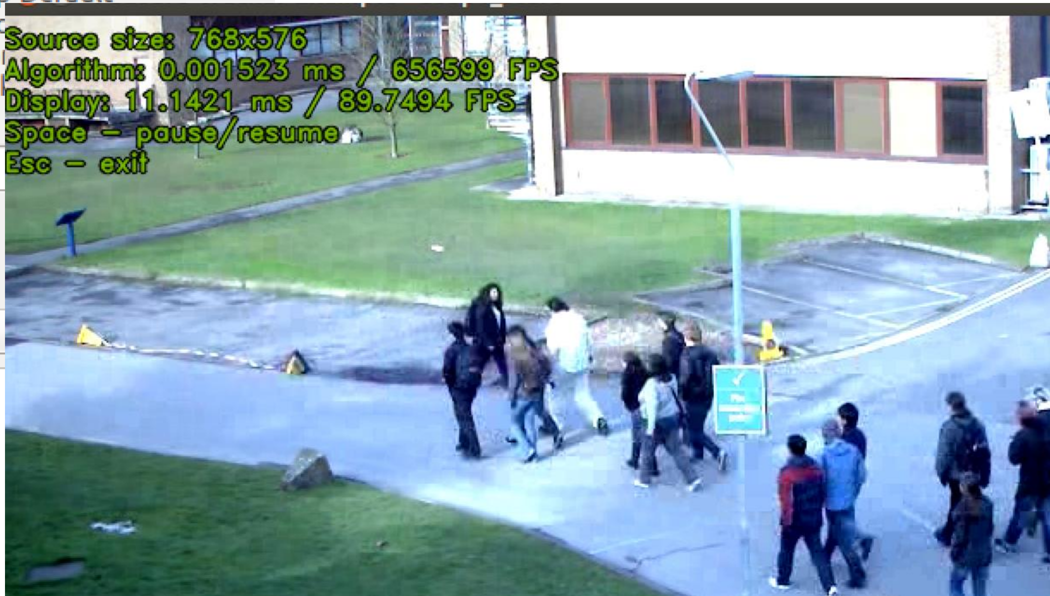
The directory "/home/openvx/tutorial_sources/build-step1_start-Desktop-Default" specified in build configuration "Default", for target "Desktop" does not c .cbp file. Qt Creator needs to recreate this file by running CMake. Some pr require command line arguments to the initial CMake call. Note that CMal remembers command line arguments from the previous runs.

Arguments:

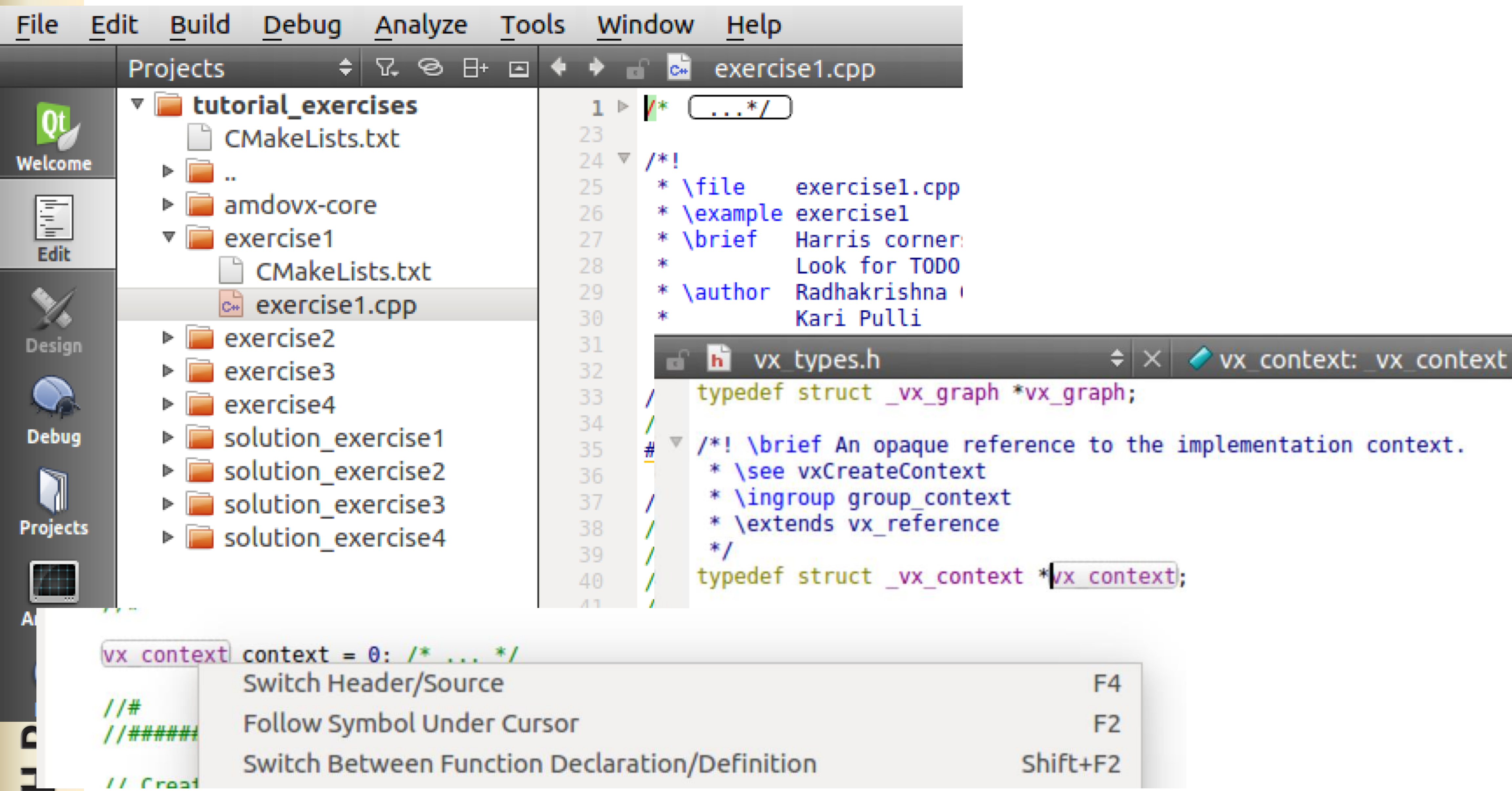
Generator: Unix Generator (Desktop)

```
-- The C compiler identification is GNU 4.8.4
-- The CXX compiler identification is GNU 4.8.4
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
```





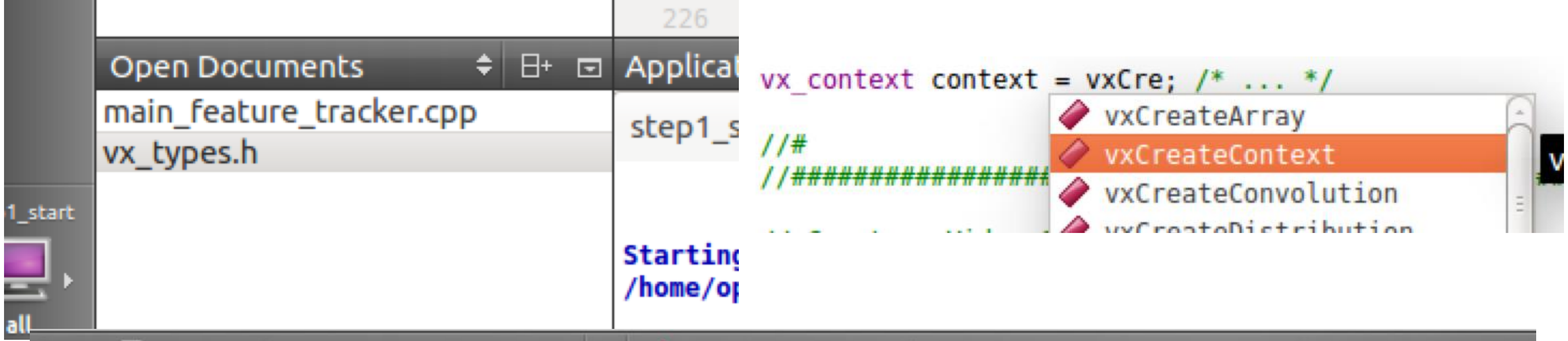
```
Source size: 768x576
Algorithm: 0.001523 ms / 656599 FPS
Display: 11.1421 ms / 89.7494 FPS
Space - pause/resume
Esc - exit
```



```
/* ...*/

/*!
 * \file     exercise1.cpp
 * \example exercise1
 * \brief   Harris corner
 *          Look for TODO
 * \author  Radhakrishna
 *          Kari Pulli
```

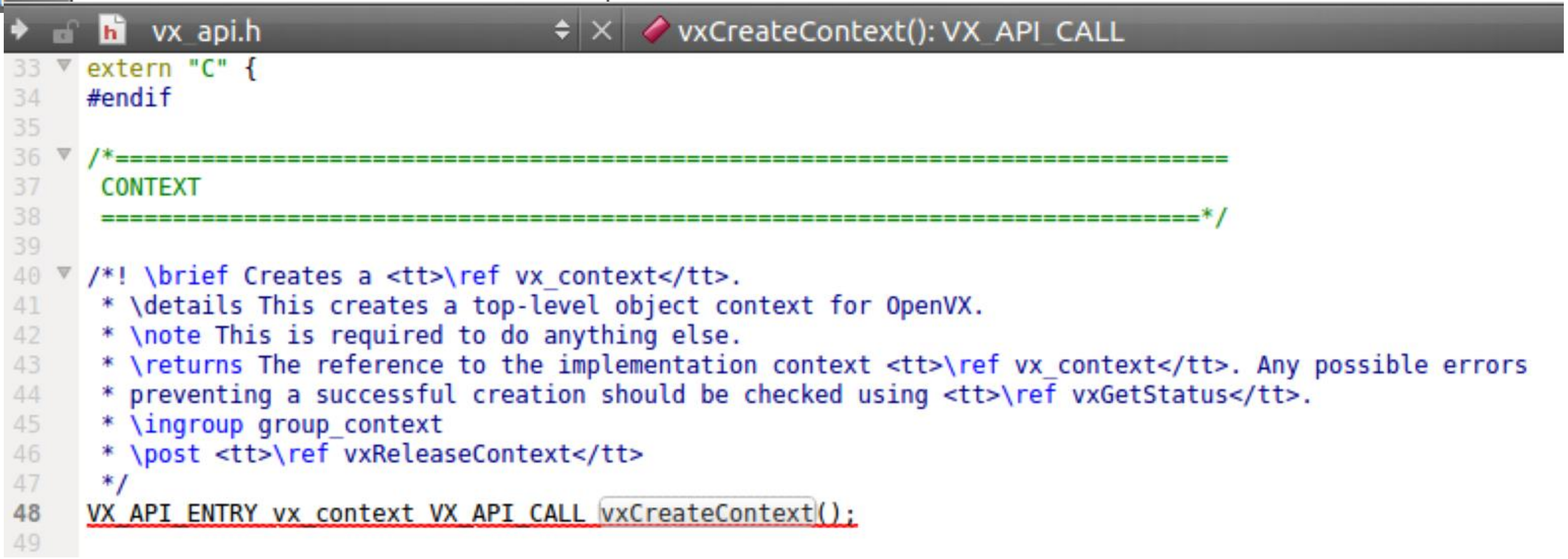
```
typedef struct _vx_graph *vx_graph;

/*! \brief An opaque reference to the implementation context.
 * \see vxCreateContext
 * \ingroup group_context
 * \extends vx_reference
 */
typedef struct _vx_context *vx_context;
```

```
vx_context context = 0; /* ... */

//#
//######

// Creat
```



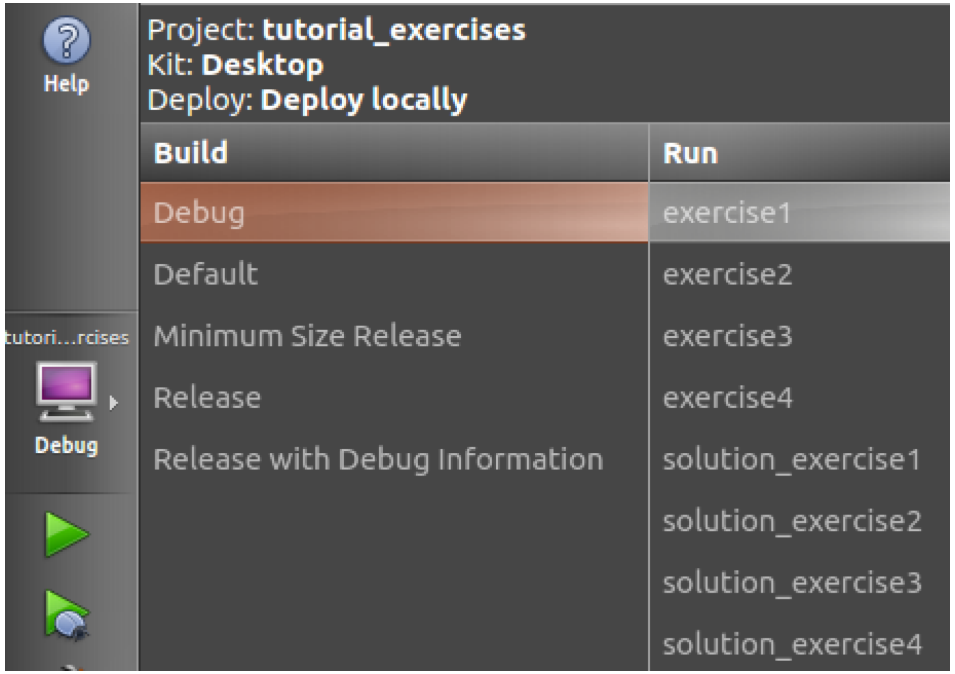
```
vx_context context = vxCre; /* ... */
//#
//################
```

```
Starting
/home/op
```



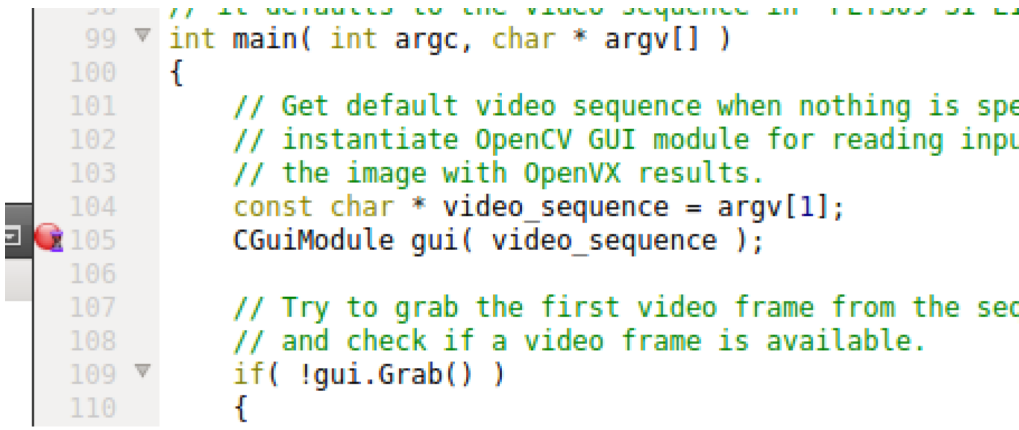
```
extern "C" {
#endif

/*=============================================================================
 CONTEXT
 ============================================================================*/

/*! \brief Creates a <tt>\ref vx_context</tt>.
 * \details This creates a top-level object context for OpenVX.
 * \note This is required to do anything else.
 * \returns The reference to the implementation context <tt>\ref vx_context</tt>. Any possible errors
 * preventing a successful creation should be checked using <tt>\ref vxGetStatus</tt>.
 * \ingroup group_context
 * \post <tt>\ref vxReleaseContext</tt>
 */
VX_API_ENTRY vx_context VX_API_CALL vxCreateContext();
```

CTRL-T

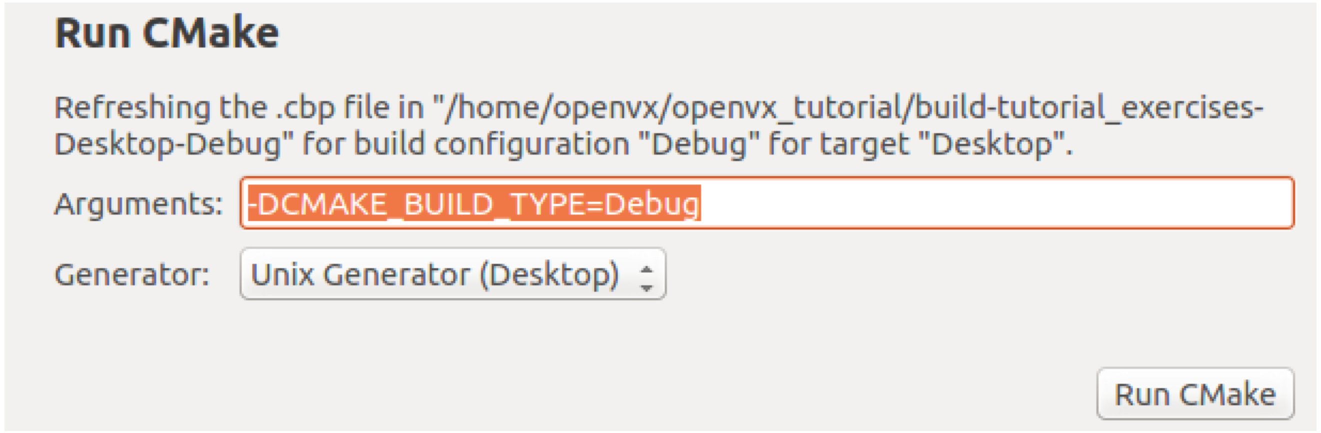


Project: **tutorial_exercises**
Kit: **Desktop**
Deploy: **Deploy locally**

Build	Run
Debug	exercise1
Default	exercise2
Minimum Size Release	exercise3
Release	exercise4
Release with Debug Information	solution_exercise1
	solution_exercise2
	solution_exercise3
	solution_exercise4



```
int main( int argc, char * argv[] )
{
    // Get default video sequence when nothing is spe
    // instantiate OpenCV GUI module for reading inpu
    // the image with OpenVX results.
    const char * video_sequence = argv[1];
    CGuiModule gui( video_sequence );

    // Try to grab the first video frame from the seq
    // and check if a video frame is available.
    if( !gui.Grab() )
    {
```

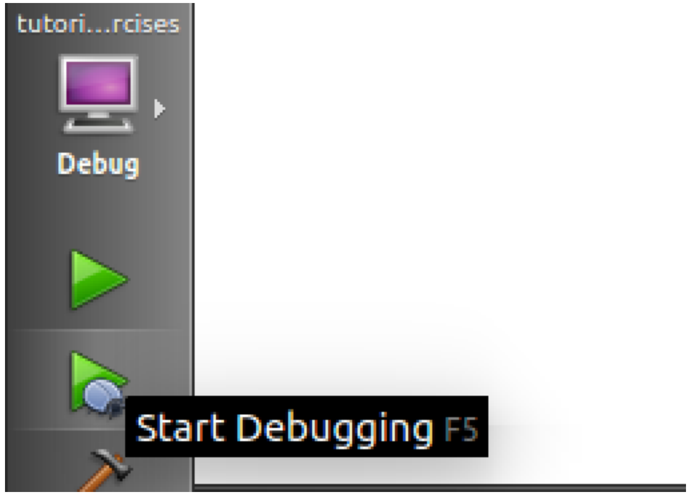


Run CMake

Refreshing the .cbp file in "/home/openvx/openvx_tutorial/build-tutorial_exercises-Desktop-Debug" for build configuration "Debug" for target "Desktop".

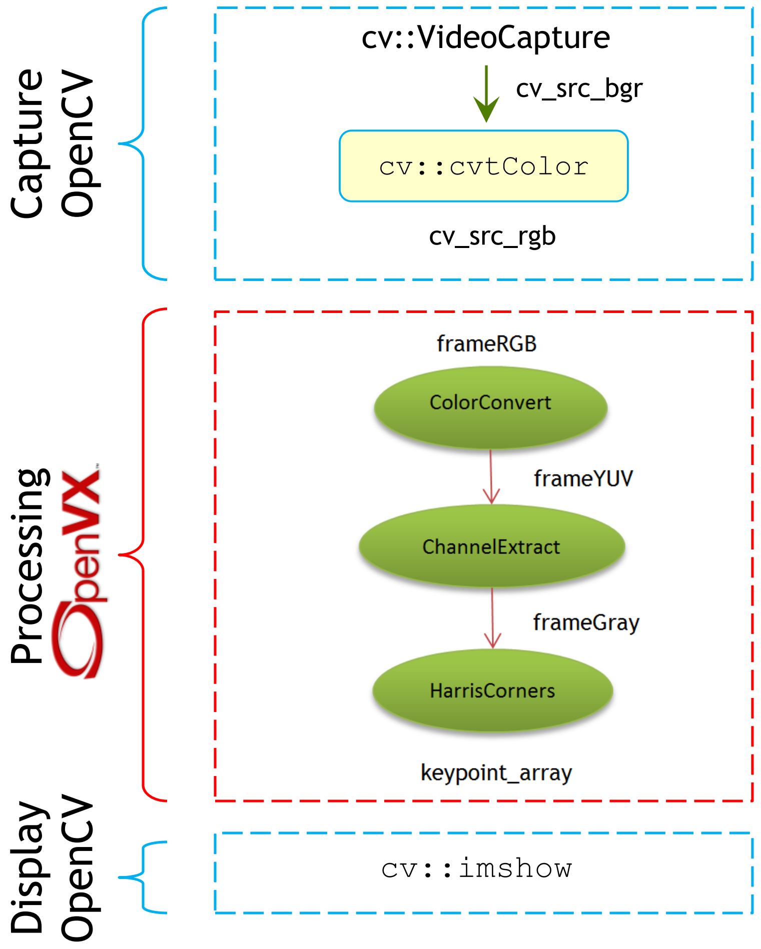
Arguments: -DCMAKE_BUILD_TYPE=Debug

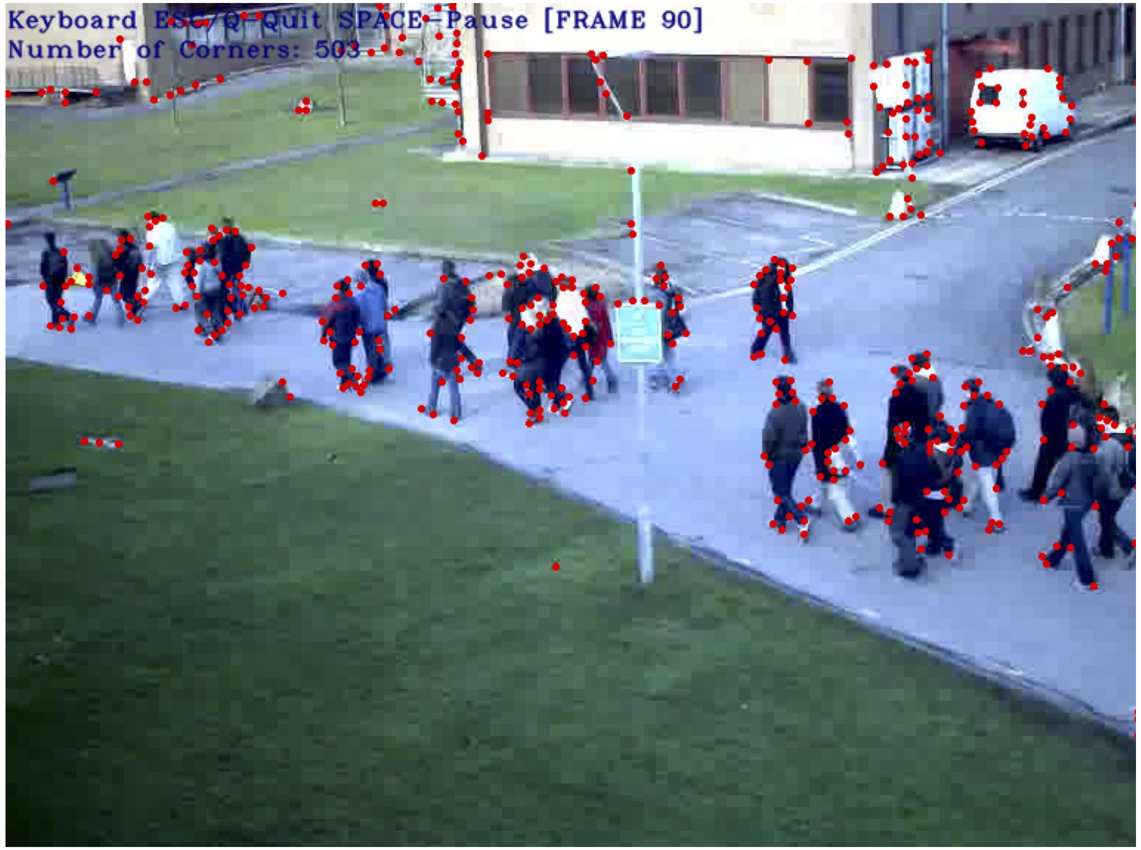
Generator: Unix Generator (Desktop)



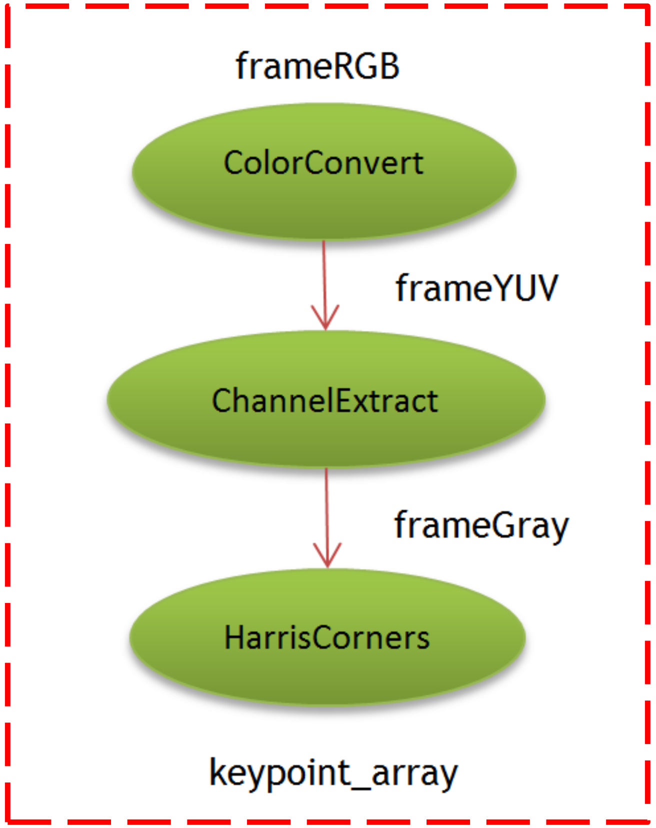
Step 1: Keypoint detection

PETS09-S1-L1-View001.avi




```
Keyboard ESC/Q-Quit SPACE-Pause [FRAME 90]
Number of Corners: 503
```

Step 1: OpenVX Concepts



- **The world**
  - vx_context
- **Error management**
- **Data object**
  - vx_image, vx_array, vx_scalar
  - Creation / Release
  - Read and write access
- **Vision functions**
  - Immediate execution mode
  - Retained execution mode (graph)

Context

- **Context**
  - OpenVX world: need to be created first
  - All objects belong to a context

```
vx_context context = vxCreateContext();
```

Error Management

- **Methods return a status**
  - `vx_status` **returned:** `VX_SUCCESS` **when no error**

```
if( vxuColorConvert( context, input, output ) != VX_SUCCESS) { /* Error */ }
```

- **Explicit status check**
  - **Object creation: use** `vxGetStatus` **to check the object**

```
vx_context context = vxCreateContext();
if( vxGetStatus( (vx_reference)context) != VX_SUCCESS ) { /* Error */ }
```

- **More info from the log callback**

```
void logCallback( vx_context c, vx_reference r, vx_status s,
                  const vx_char string[] )
{ /* Do something */ }
...

vxRegisterLogCallback( context, logCallback, vx_false_e );
```

Data objects

- **The application gets only references to objects, not the objects**
  - References should be released by the application when not needed
  - Ref-counted object destroyed by OpenVX when not referenced any more

```
vx_image img = vxCreateImage( context, 640, 400, VX_DF_IMAGE_RGB );
// Use the image
vxReleaseImage( &img );
```

- **Object-Oriented Behavior**
  - strongly typed (good for safety-critical applications)
  - OpenVX are really pointers to structs
    - any object may be down-cast to a vx_reference, e.g., for passing to vxGetStatus()
- **Opaque**
  - Access to content explicit and temporary (access, edit, commit)
    - No permanent pointer to internal data
  - Needed to handle complex memory hierarchies
    - DSP local memory
    - GPU dedicated memory

Image Access (1/3) : Overview

- **Access limited in time**
  - vxAccessImagePatch: get access (Read, Write, Read & Write)
  - vxCommitImagePatch: release the access
- **Two modes**
  - MAP: <u>OpenVX</u> controls *address* and *memory layout*

```
void * ptr = NULL;
vx_imagepatch_addressing_t addr;
vx_rectangle_t rect = { 0u, 0u, width, height };
vxAccessImagePatch( img, &rect, plane, &addr, &ptr, VX_READ_AND_WRITE );
// Access data in ptr
vxCommitImagePatch( img, &rect, plane, &addr, ptr );
```

  - COPY: The <u>application</u> controls *address* and *memory layout*

```
void * ptr = &my_array[0];
vx_imagepatch_addressing_t addr = { /* to fill */ };
vx_rectangle_t rect = { 0u, 0u, width, height };
vxAccessImagePatch( img, &rect, plane, &addr, &ptr, VX_READ_AND_WRITE );
// Access data in my_array
vxCommitImagePatch( img, &rect, plane, &addr, ptr );
```

Image Access (2/3) : Patch

```
typedef struct _vx_rectangle_t {
    vx_uint32 start_x;          /*!< \brief The Start X coordinate. */
    vx_uint32 start_y;          /*!< \brief The Start Y coordinate. */
    vx_uint32 end_x;            /*!< \brief The End X coordinate. */
    vx_uint32 end_y;            /*!< \brief The End Y coordinate. */
} vx_rectangle_t;
```

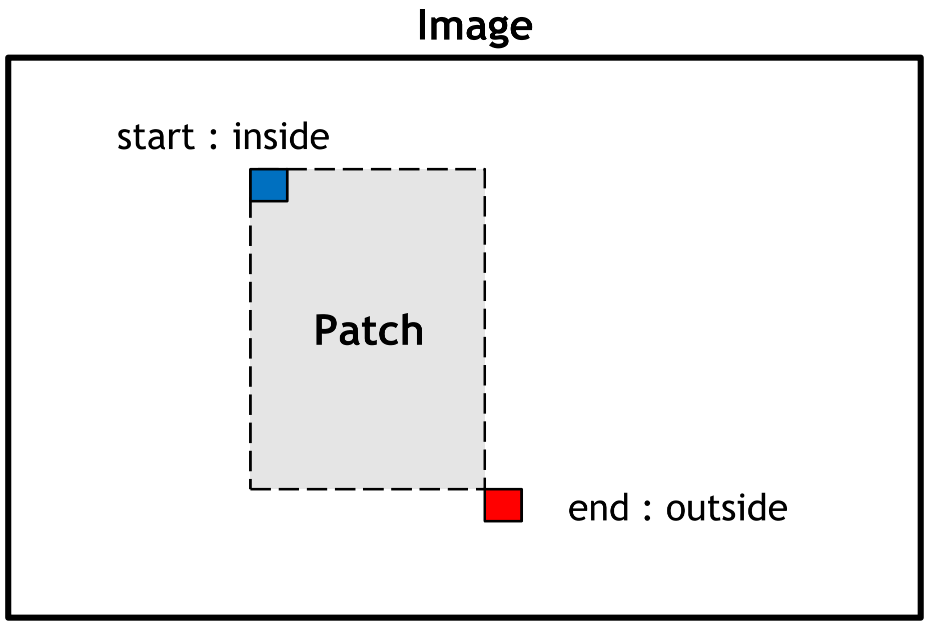
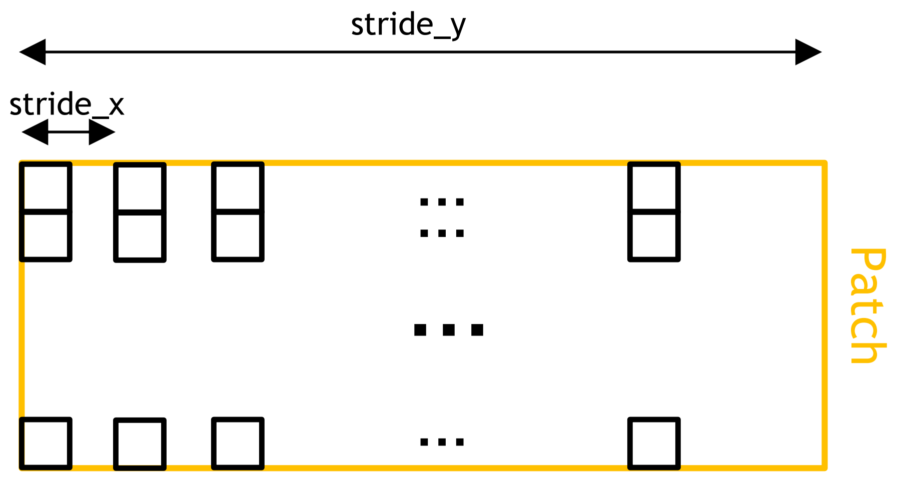


Image Access (3/3) : Memory Layout

```
typedef struct _vx_imagepatch_addressing_t {
    vx_uint32 dim_x;
    vx_uint32 dim_y;
    vx_int32  stride_x;
    vx_int32  stride_y;
    vx_uint32 scale_x;
    vx_uint32 scale_y;
    vx_uint32 step_x;
    vx_uint32 step_y;
} vx_imagepatch_addressing_t;
```

- dim_x: Num of (logical) pixels in a row
- dim_y: Num of (logical) pixels in a column
- stride_x: Num of bytes between the beginning of 2 successive pixels
- stride_y: Num of bytes between the beginning of 2 successive lines

scale_x, scale_y, step_x, step_y: Sub-sampling :
1 *physical* pixel every 'step' *logical* pixel
scale = VX_SCALE_UNITY / step



Miscellaneous

- Array
  - Variable number of elements, but fixed maximum capacity

```
vx_array array = vxCreateArray( context, VX_TYPE_KEYPOINT, 10000 );
```

  - Access philosophy is similar to the image (MAP / COPY)

```
vx_size num;
vxQueryArray( array, VX_ARRAY_ATTRIBUTE_NUMITEMS, &num, sizeof(num) );

vx_keypoint_t * ptr = NULL; // access in MAP mode
vx_size stride;
vxAccessArrayRange( array, 0, num, &stride, (void **)&ptr, VX_READ_ONLY );
/* Access */
vxCommitArrayRange( array, 0, num, ptr );
```

- Scalar

```
vx_float32 distance  = 5.f;
vx_scalar s_distance = vxCreateScalar( context, VX_TYPE_FLOAT32, &distance );
```

Vision Functions: Immediate Execution Mode

- RGB -> YUV

```
vxuColorConvert( context, frameRGB, frameYUV );
```

```
                   VX_DF_IMAGE_RGB   VX_DF_IMAGE_YUV
```

- YUV -> Y

```
vxuChannelExtract( context, frameYUV, VX_CHANNEL_Y, frameGray );
```

```
                         VX_DF_IMAGE_YUV                VX_DF_IMAGE_U8
```

- Harris corner

- strength_thresh	: 0.0005f
- min_distance	: 5.0f
- sensitivity	: 0.04f
- gradient_size	: 3
- block_size	: 3

```
vxuHarrisCorners( context, frameGray, s_strength_thresh, s_min_distance,
                  s_k_sensitivity, gradientSize, blockSize,
                  keypoint_array, NULL );
```