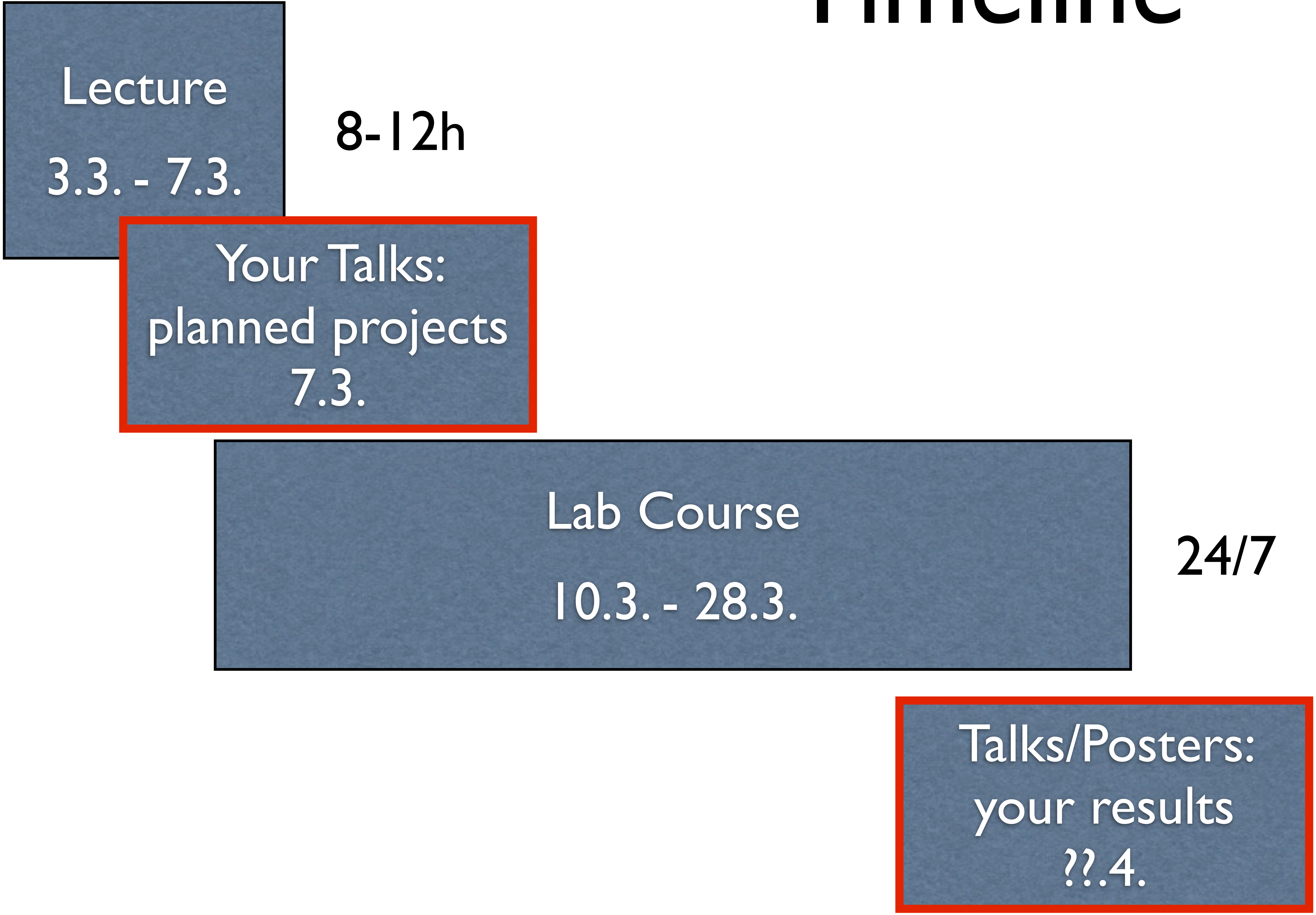
NVIDIA

CUDA™
TEACHING
CENTER

Göttingen

Timeline



Lecture

3.3. - 7.3.

8-12h

Your Talks:
planned projects
7.3.

Lab Course

10.3. - 28.3.

24/7

Talks/Posters:
your results
??.4.

Goals

- learn CUDA
- learn to use CUDA-based libraries
- use CUDA in a C++ environment
- convert project into article (cf. step-16)

Modules

B.Mat **106**
Grundlagen Wiss. Rechnen

B.Mat.**730**
Weiterführung Praktikum
Wiss. Rechnen

BSc.
Computer
Science

BSc.
Math,
Physics

Masters: ?

Projects

Each participant works on a larger project which he/she presents before the lab course starts. The talk should be about 15-20 minutes and cover:

- what will be simulated
- which part of the problem is supposed to profit from parallelization
- sketch of algorithm
- algorithmic problems due to parallelization (some algorithms simply cannot be parallelized)

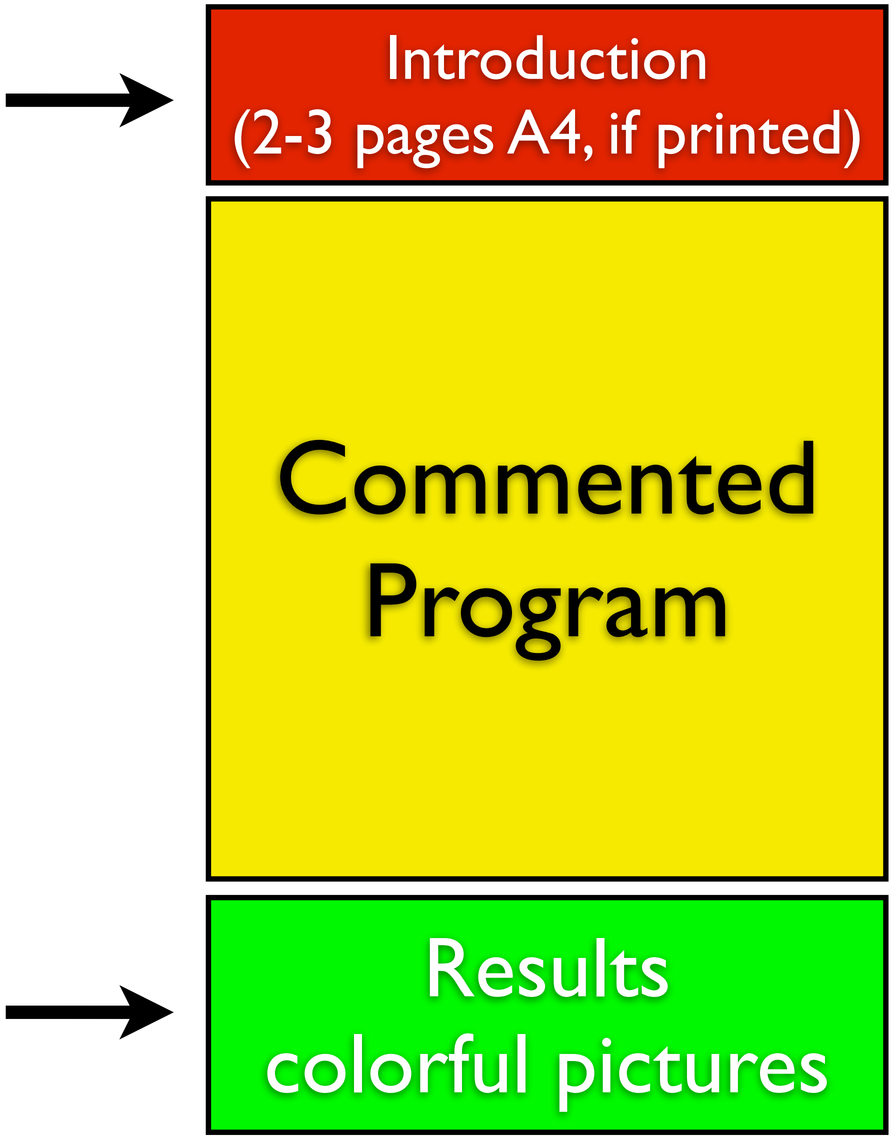
- quantum time propagation
- matrix assembly for integral equations (FEM-BEM coupling)
- super-resolution algorithms for optical microscopy
- X-ray physics (all sorts of phase reconstruction)

Example: 2013's Lecture

	Mo	Di	Mi	Do	Fr
8:15 - 9:45	**CUDA Basics** programming model, hardware, memory transfers, C++	**CUDA Advanced** more memory handling, OpenGL, PAAL	**Fluid Dynamics** *A. Tilgner*	**Finite Differences Multigrid** theory, implementation	Rashmi Barbate Prefix Sums Sanjeev Laha Histograms Simon Maretzke QM Time Propagation break [illegible] GPU-assisted routing
10:00 - 11:30	**Linear Algebra** sparse matrices, deal.II interoperability	**C++ Expression Templates** SciPal, dense LA	**Finite Elements C(ontinuous)** *G. Lube*	**Boundary Elements** algorithms and implementation	Moritz Doll Searching/Sorting Simon Schütz R[illegible]v. break Sabyasachi Gosh Histograms Pranay Tare Preconditioning / Multigrid
					Christian Holme BEM Assembly
12:00 - 14:30	**Project Structure** svn, documenting, QTCreator, coding conventions	**Debugging** Tools for analysing CUDA code **OpenMP**	**QThreads**		break

Lab Course

Structure of programs:



This will be the essence of your introductory talk

Introduction
(2-3 pages A4, if printed)

Commented Program

Results
colorful pictures

This will be the **basis** for your final talk/poster

- Core time: 9-16 h
- Show up each day for discussions, etc ...
- Stick to the prepared structure of the steps

Further details:

http://num.math.uni-goettingen.de/~stkramer/doc/tutorial/index.html

Presentation of Results

- to be scheduled for late april
- either seminar or poster session, depends on the number of participants

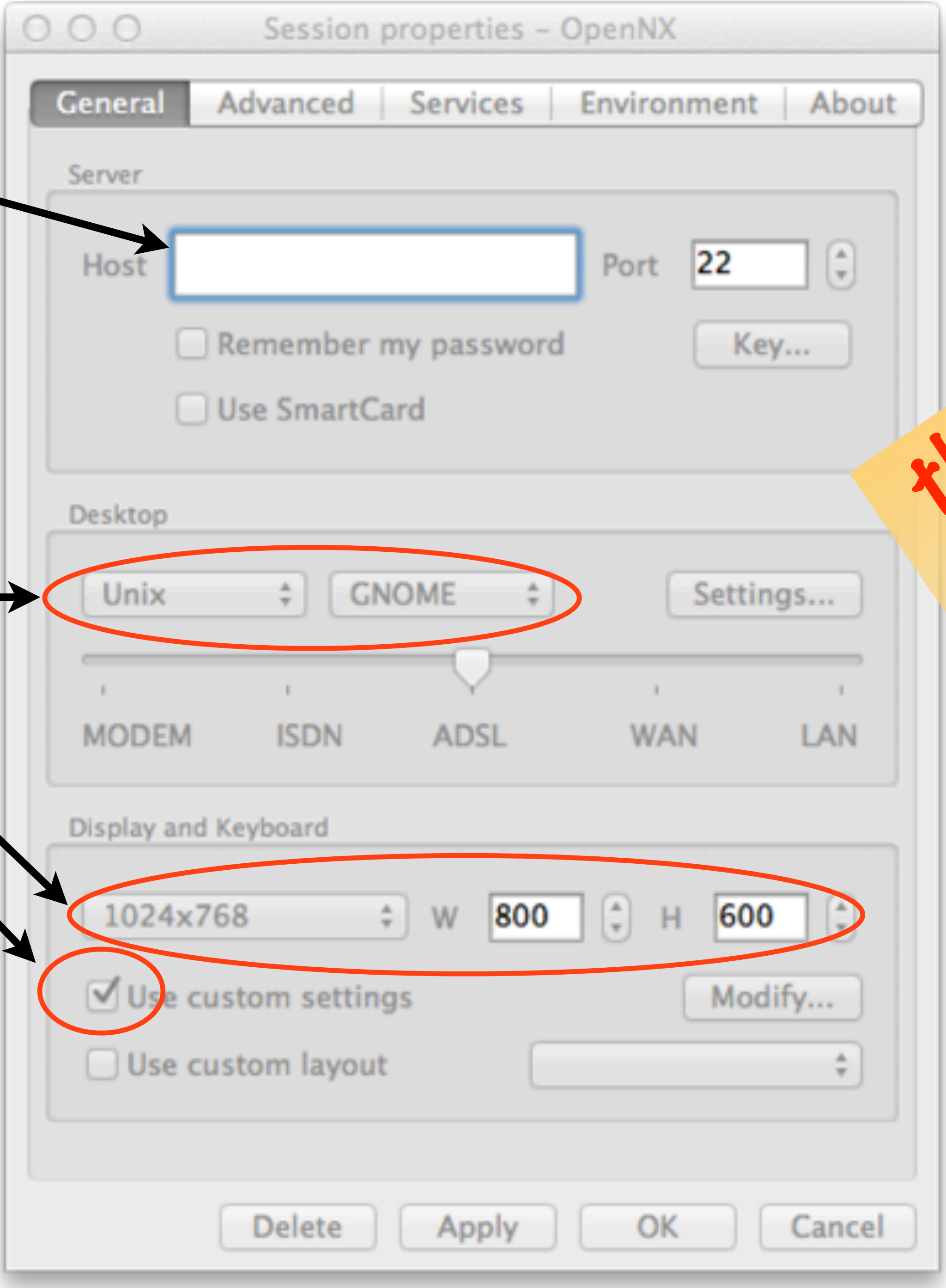
Lab Course

How To

Login via NXClient/OpenNX

enter the IP I emailed you

window manager config as shown here



Session properties – OpenNX

Server

Host

Port 22

☐ Remember my password

☐ Use SmartCard

Desktop

Unix GNOME

Display and Keyboard

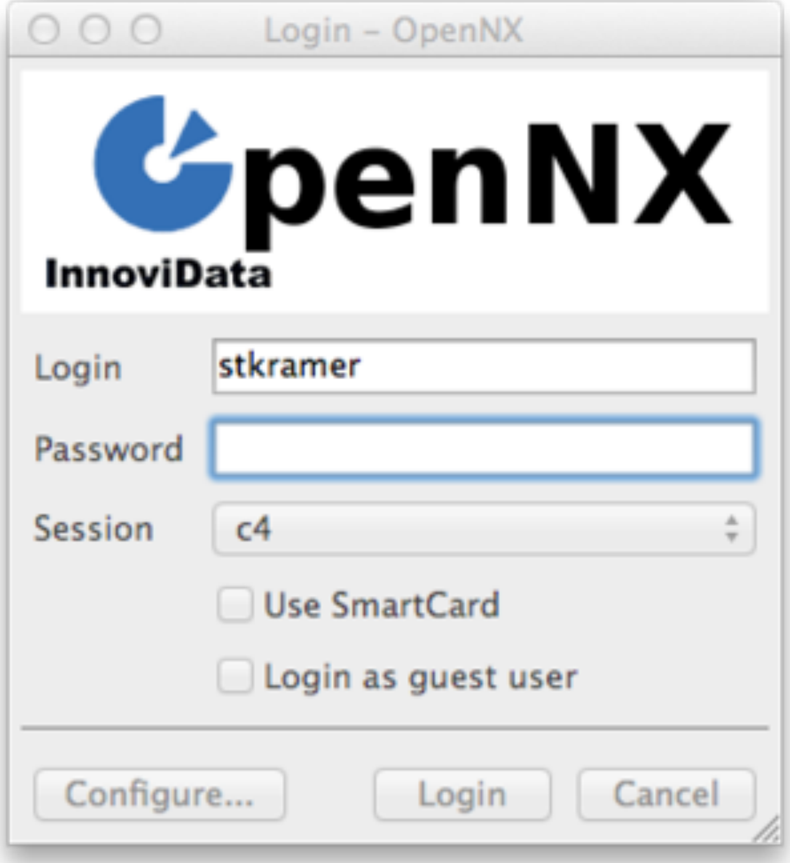
1024x768 W 800 H 600

☑ Use custom settings

☐ Use custom layout

this still works, but x2go is the preferred way. Works essentially the same way

Then: log in



Login – OpenNX

Login stkramer

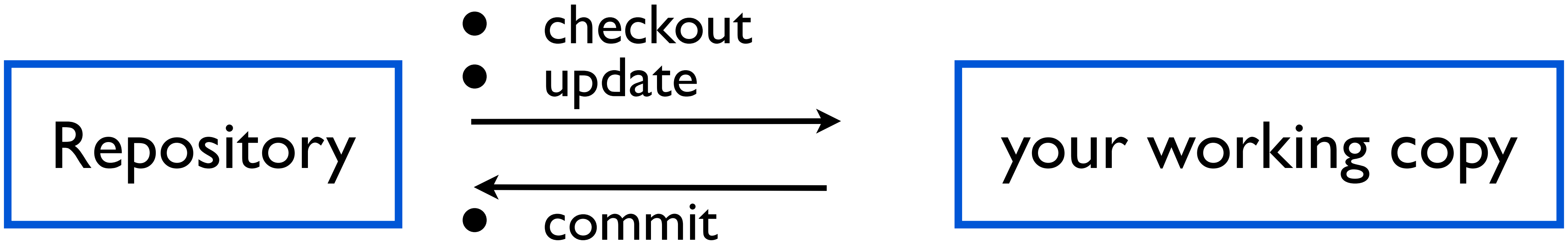
Password

Session c4

☐ Use SmartCard

☐ Login as guest user

svn



- checkout
- update

Repository

your working copy

- commit

repository location (once it exists & you got access):

https://svn.num.math.uni-goettingen.de/svn/cuda/Praktikum_2014

location of working copy: *somewhere in your home directory*

how to create it:

1. open a terminal
2. create a folder 'cuda' in your home directory:
   ```
   mkdir ~/cuda
   ```
3. go to that directory:
   ```
   cd ~/cuda
   ```
4. check out the latest version of the whole lab course:
   ```
   svn checkout https://svn.num.math.uni-goettingen.de/svn/cuda/Praktikum_2014
   ```
5. once you have changed something which you want to keep, commit it to the repository:
   ```
   cd ~/cuda/Praktikum_2014/<some subdir>
   
   svn commit -m "<my commit message>"
   ```

At some point you have to enter your account password for the svn access

Your Project

location :

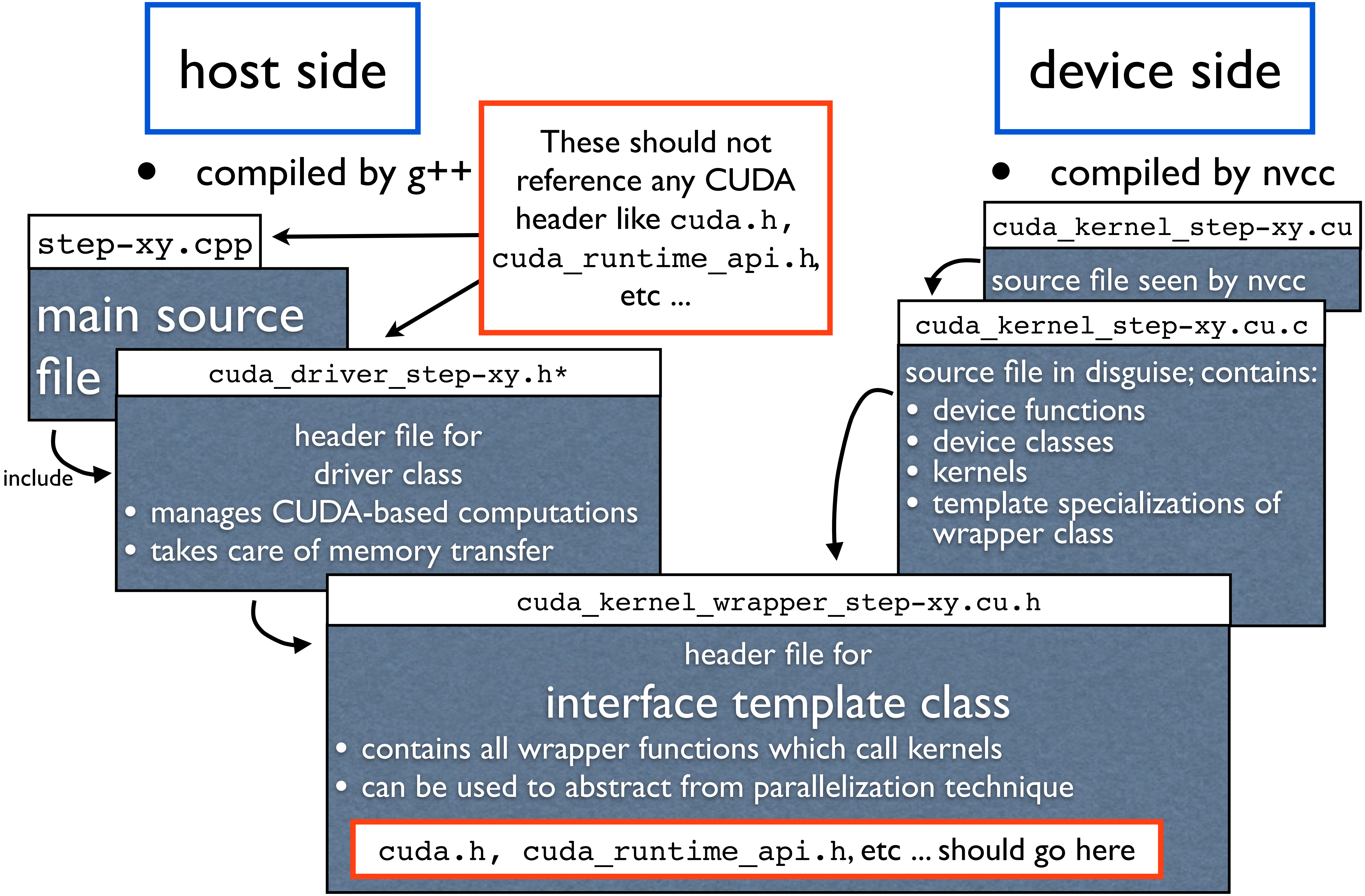
```
Praktikum_2014/testsite/step-<xy>
```

`<xy>` will be some number

central configuration file:

```
step-<xy>/step-<xy>.pro
```

Project Structure



host side

- compiled by g++

These should not reference any CUDA header like cuda.h, cuda_runtime_api.h, etc ...

device side

- compiled by nvcc

step-xy.cpp

main source file

include

cuda_driver_step-xy.h*

header file for driver class

- manages CUDA-based computations
- takes care of memory transfer

cuda_kernel_step-xy.cu

source file seen by nvcc

cuda_kernel_step-xy.cu.c

source file in disguise; contains:

- device functions
- device classes
- kernels
- template specializations of wrapper class

cuda_kernel_wrapper_step-xy.cu.h

header file for

interface template class

- contains all wrapper functions which call kernels
- can be used to abstract from parallelization technique

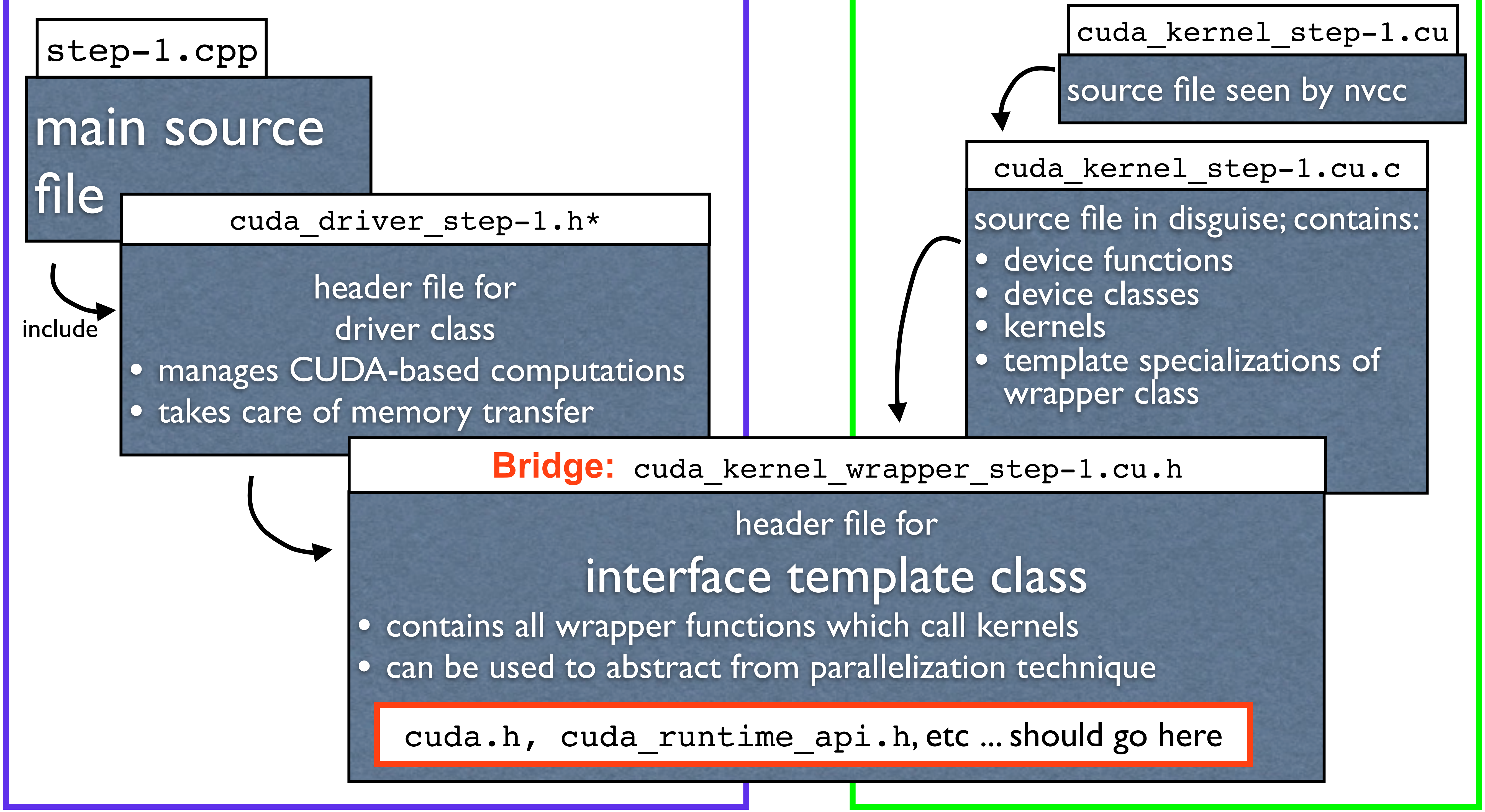
cuda.h, cuda_runtime_api.h, etc ... should go here

host side

- compiled by g++

device side

- compiled by nvcc



step-1.cpp

main source file

include

cuda_driver_step-1.h*

header file for

driver class

- manages CUDA-based computations
- takes care of memory transfer

cuda_kernel_step-1.cu

source file seen by nvcc

cuda_kernel_step-1.cu.c

source file in disguise; contains:

- device functions
- device classes
- kernels
- template specializations of wrapper class

Bridge: cuda_kernel_wrapper_step-1.cu.h

header file for

interface template class

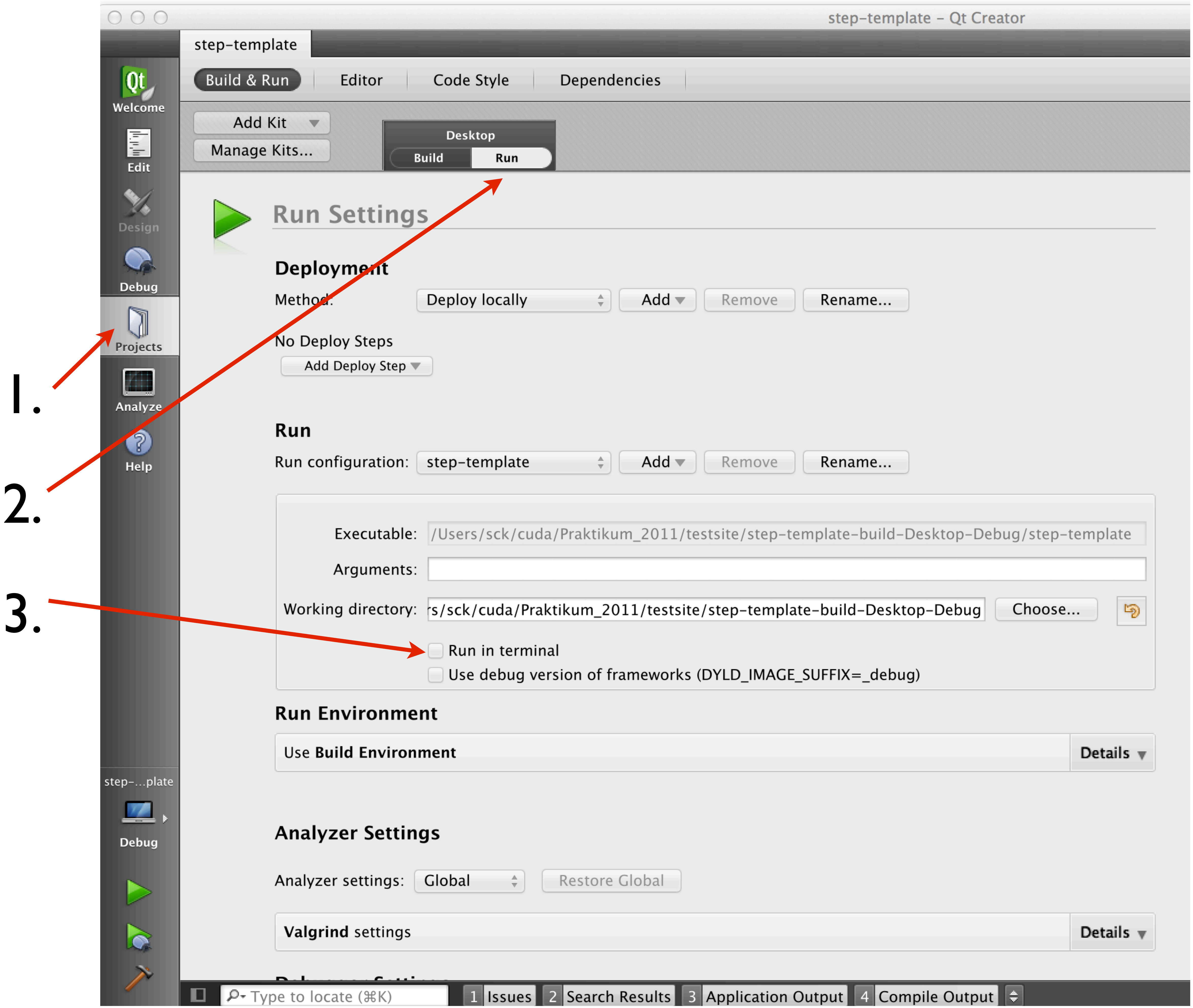
- contains all wrapper functions which call kernels
- can be used to abstract from parallelization technique

cuda.h, cuda_runtime_api.h, etc ... should go here

QtCreator

- can be found in Ubuntu's Application/Developer menu
- to work on your project, open `step-<xy>.pro`
- the "Shadow build" configuration can be found in the project settings of QtCreator
- **to avoid lengthy paths for the shadow build directories change the setting in**

  Preferences/Build&Run/Default Build Directory
- to run your program from within QtCreator, disable "run in terminal" (see next page)



Run Settings

Deployment

Method: Deploy locally | Add | Remove | Rename...

No Deploy Steps

Add Deploy Step

Run

Run configuration: step-template | Add | Remove | Rename...

Executable: /Users/sck/cuda/Praktikum_2011/testsite/step-template-build-Desktop-Debug/step-template

Arguments:

Working directory: rs/sck/cuda/Praktikum_2011/testsite/step-template-build-Desktop-Debug | Choose...

□ Run in terminal

□ Use debug version of frameworks (DYLD_IMAGE_SUFFIX=_debug)

Run Environment

Use **Build Environment** | Details

Analyzer Settings

Analyzer settings: Global | Restore Global

Valgrind settings | Details

Generate Project’s Doc

go to :

```
Praktikum_2014/scripts/doc
```

run script:

```
./make_step_doc.py <xy>
```

`<xy>` is the number of your project

open in a browser:

```
firefox ../../doc/tutorial/index.html
```

... and look for your number in the navigation bar on the left

for further details:

```
Praktikum_2014/readme.html
```