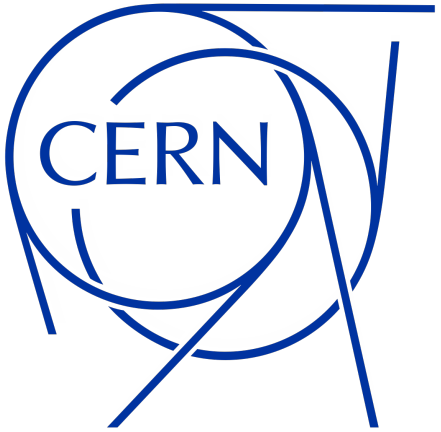


Short Term Internship Report

Adrian Düsselberg

02.09.2024

Introduction

- **Adrian**
- **Bachelor student in Engineering Science from the Technical University of Munich**
- **6 month short term internship (started in April)**
- **ROOT, Automated ROOT Graphic Testing Suite, ROOT Doxygen Doc (Bonus)**

ROOT

Data Analysis Framework

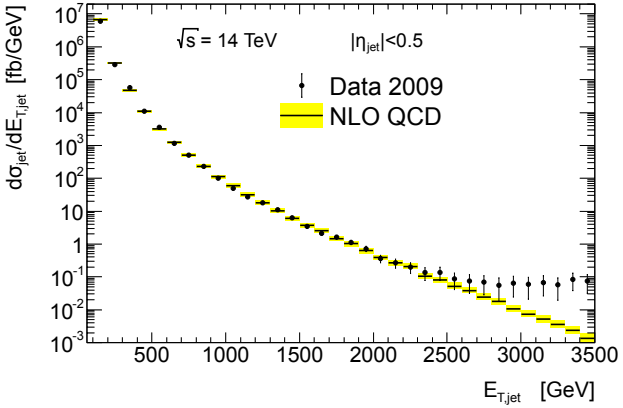
Outline

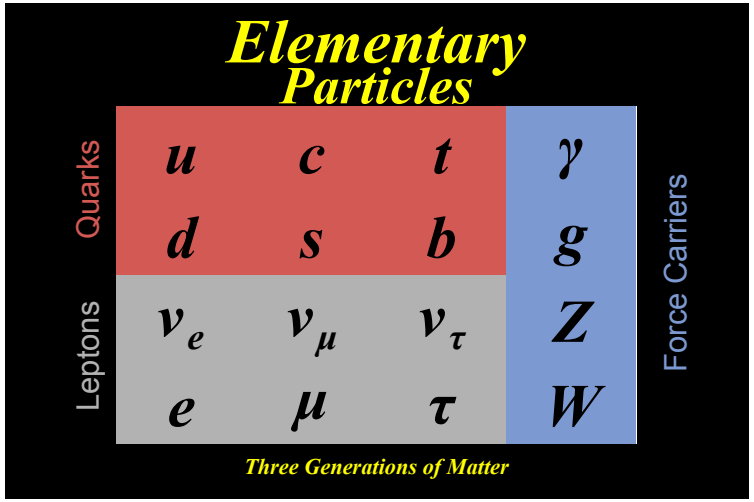
- ROOT Graphic Testing Suite
- ROOT Doxygen Documentation (Bonus)
- Personal Conclusion

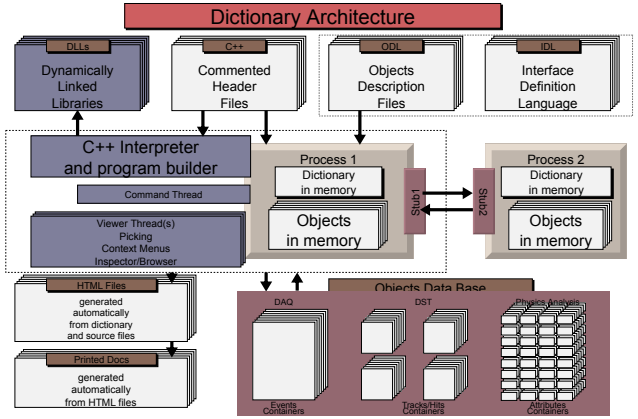
ROOT Graphic testing suite

Motivation

- **Why ROOT web graphics should be tested!?**
  - Web graphics will be the default in the future
  - Tests are necessary to ensure reliable graphics for users
- **ROOT POW: Visualisation and UI**
  - Automate web-based graphic test suite

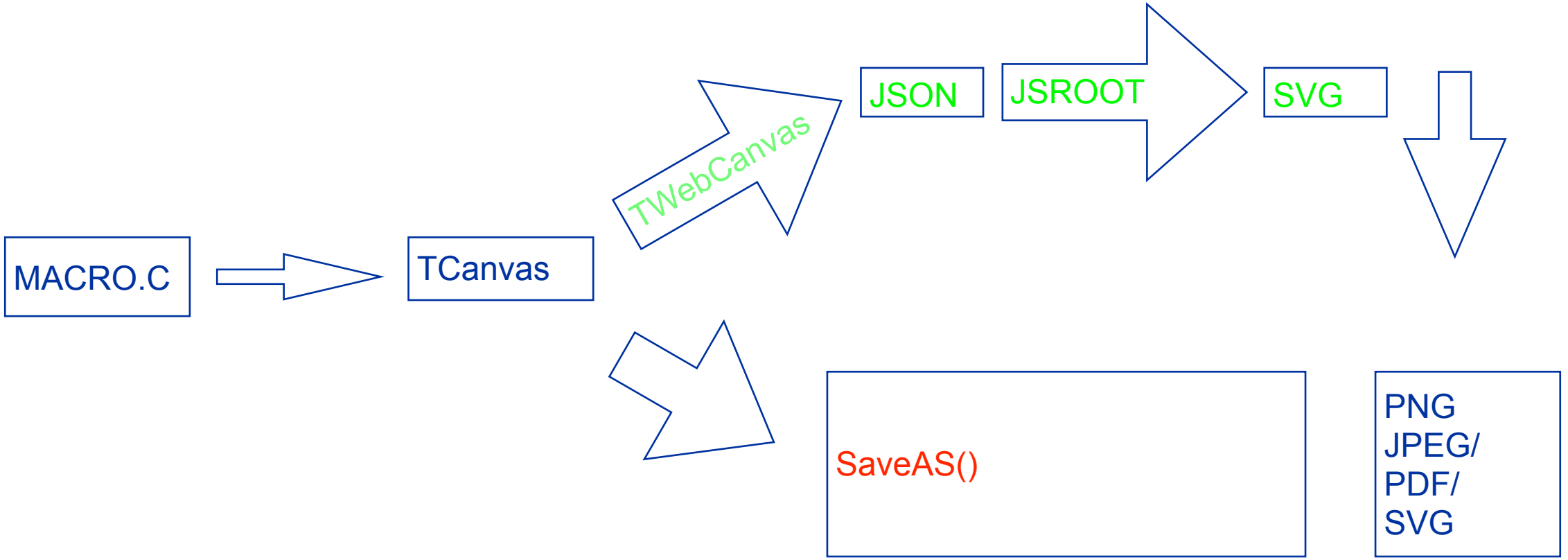






ROOT Graphics

- **Old vs. New (Web) Graphics?**



Objectives

- **Provide GitHub CI for JSROOT**
  - 3D images produces with headless-gl can differ on differnet platforms -> standard platform
- **Provide an automated ROOT (WEB) Graphic Test**
  - Collect and organize all relevant macros for TCanvas/JSON generations
  - Describe corner cases where TWEBCanvas does not work/ produce the same results
    - Fix them or precisely document them
  - Integrate the battery of tests into ROOT GitHub CI

Objectives

- **Provide GitHub CI for JSROOT**
  - 3D images produces with headless-gl can differ on differnet platforms -> standard platform
- **Provide an automated ROOT (WEB) Graphic Test**
  - Collect and organize all relevant macros for TCanvas/JSON generations
  - Describe corner cases where TWEBCanvas does not work/ produce the same results
    - Fix them or precisely document them
  - Integrate the battery of tests into ROOT GitHub CI

Warm Up! JSROOT and JSROOT-TEST (Serguey)

- **JSROOT-TEST tests basic functionalties of JSROOT, using Node.js**
  1. Executed jsroot-test with jsroot-ci on different platforms
  2. Included svg comparsion in JSROOT-TEST
  3. Debugged tests and excluded non reproducable parts, like image tags
  4. Introduced a standard platform and a list of special tests
  5. Merged changes into jsroot and jsroot-test (Serguey)

- **Result: JSROOT-TEST is included in GitHub CI of JSROOT on one standard platform (ubuntu)**
- **JSROOT-TEST: https://github.com/linev/jsroot-test**

Objectives

- **Provide GitHub CI for JSROOT**
  - 3D images produces with headless-gl can differ on differnet platforms -> standard platform
- **Provide an automated ROOT (WEB) Graphic Test**
  - Collect and organize all relevant macros for TCanvas/JSON generations
  - Describe corner cases where TWEBCanvas does not work/ produce the same results
    - Fix them or precisely document them
  - Integrate the battery of tests into ROOT GitHub CI

ROOT-Graphic-Test: Architecture & Functionality

- **As usual you start from a private Repo: https://github.com/DuesselbergAdrian/root-graphic-tests**

```
root-graphic-test/
├── macros/
│   ├── graphics/
│   ├── graphs/
│   ├── hist/
│   └── ....
├── json_ref/
├── new_svg_ref/
├── pdf_ref/
├── old_svg_ref/
│
├── package.json
│
├── CMAKELists.txt
├── macros_list.json
├── generate_macros_cmake.py
│
├── Test_Root.cpp
├── Test_JsRoot.js
│
├── README.md
├── CONTRIBUTING.md
├── ARCHITECTURE.md
└── LICENSE.md
```

1. Add Test with Cmake and run Ctest
2. Modified Macro (macros)
3. Test files
   1. Generate files according to test type
      - Old SVG
      - PDF
      - JSON
      - New SVG (node.js, Test_JsRoot.js)
   2. Preprocess the file content
   3. Compare generated files to references
4. Output the test result

ROOT-Graphic-Test: Architecture & Functionality

- **Repo: https://github.com/DuesselbergAdrian/root-graphic-tests**

```
root-graphic-test/
├── macros/
│   ├── graphics/
│   ├── graphs/
│   ├── hist/
│   └── ....
├── json_ref/
├── new_svg_ref/
├── pdf_ref/
├── old_svg_ref/
│
├── package.json
│
├── CMAKELists.txt
├── macros_list.json
├── generate_macros_cmake.py
│
├── Test_Root.cpp
├── Test_JsRoot.js
│
├── README.md
├── CONTRIBUTING.md
├── ARCHITECTURE.md
└── LICENSE.md
```

1. Add Test with Cmake and run Ctest
2. Modified Macro (macros)
3. Test files
   1. Generate files according to test type
      - Old SVG
      - PDF
      - JSON
      - New SVG (node.js, Test_JsRoot.js)
   2. Preprocess the file content
   3. Compare generated files to references
4. Output the test result

ROOT-Graphic-Test: Architecture & Functionality

- **Repo: https://github.com/DuesselbergAdrian/root-graphic-tests**

```
root-graphic-test/
├── macros/
│   ├── graphics/
│   ├── graphs/
│   ├── hist/
│   └── ....
├── json_ref/
├── new_svg_ref/
├── pdf_ref/
├── old_svg_ref/
│
├── package.json
│
├── CMAKELists.txt
├── macros_list.json
├── generate_macros_cmake.py
│
├── Test_Root.cpp
├── Test_JsRoot.js
│
├── README.md
├── CONTRIBUTING.md
├── ARCHITECTURE.md
└── LICENSE.md
```

1. Add Test with Cmake and run Ctest
2. Modified Macro (macros)
3. Test files
   1. Generate files according to test type
      - Old SVG
      - PDF
      - JSON
      - New SVG (node.js, Test_JsRoot.js)
   2. Preprocess the file content
   3. Compare generated files to references
4. Output the test result

ROOT-Graphic-Test: Architecture & Functionality

- **Repo: https://github.com/DuesselbergAdrian/root-graphic-tests**

```
root-graphic-test/
├── macros/
│   ├── graphics/
│   ├── graphs/
│   ├── hist/
│   └── ....
├── json_ref/
├── new_svg_ref/
├── pdf_ref/
├── old_svg_ref/
│
├── package.json
│
├── CMAKELists.txt
├── macros_list.json
├── generate_macros_cmake.py
│
├── Test_Root.cpp
├── Test_JsRoot.js
│
├── README.md
├── CONTRIBUTING.md
├── ARCHITECTURE.md
└── LICENSE.md
```

1. Add Test with Cmake and run Ctest
2. Modified Macro (macros)
3. Test files
   1. Generate files according to test type
      - Old SVG
      - PDF
      - JSON
      - New SVG (node.js, Test_JsRoot.js)
   2. Preprocess the file content
   3. Compare generated files to references
4. Output the test result

ROOT-Graphic-Test: Architecture & Functionality

- **Repo: https://github.com/DuesselbergAdrian/root-graphic-tests**

```
root-graphic-test/
├── macros/
│   ├── graphics/
│   ├── graphs/
│   ├── hist/
│   └── ....
├── json_ref/
├── new_svg_ref/
├── pdf_ref/
├── old_svg_ref/
│
├── package.json
│
├── CMAKELists.txt
├── macros_list.json
├── generate_macros_cmake.py
│
├── Test_Root.cpp
├── Test_JsRoot.js
│
├── README.md
├── CONTRIBUTING.md
├── ARCHITECTURE.md
└── LICENSE.md
```

1. Add Test with Cmake and run Ctest
2. Modified Macro (macros)
3. Test files
   1. Generate files according to test type
      - Old SVG
      - PDF
      - JSON
      - New SVG (node.js, Test_JsRoot.js)
   2. Preprocess the file content
   3. Compare generated files to references
4. Output the test result

ROOT-Graphic-Test: Architecture & Functionality

- **Repo: https://github.com/DuesselbergAdrian/root-graphic-tests**

```
root-graphic-test/
├── macros/
│   ├── graphics/
│   ├── graphs/
│   ├── hist/
│   └── ....
├── json_ref/
├── new_svg_ref/
├── pdf_ref/
├── old_svg_ref/
│
├── package.json
│
├── CMAKELists.txt
├── macros_list.json
├── generate_macros_cmake.py
│
├── Test_Root.cpp
├── Test_JsRoot.js
│
├── README.md
├── CONTRIBUTING.md
├── ARCHITECTURE.md
└── LICENSE.md
```

1. Add Test with Cmake and run Ctest
2. Modified Macro (macros)
3. Test files
   1. Generate files according to test type
      - Old SVG
      - PDF
      - JSON
      - New SVG (node.js, Test_JsRoot.js)
   2. Preprocess the file content
   3. Compare generated files to references
4. Output the test result

```
Test project /Users/adrianduesselberg/root-graphic-tests/build
      Start   1: AtlasExample_ALL_JSON_graphics
 1/472 Test   #1: AtlasExample_ALL_JSON_graphics .............   Passed    4.76 sec
      Start   2: AtlasExample_ALL_oldSVG_graphics
 2/472 Test   #2: AtlasExample_ALL_oldSVG_graphics ............   Passed    1.65 sec
      Start   3: AtlasExample_ALL_oldPDF_graphics
 3/472 Test   #3: AtlasExample_ALL_oldPDF_graphics ............   Passed    1.54 sec
      Start   4: AtlasExample_ALL_newSVG_graphics
```

ROOT-Graphic-Test: Overview

- **Documentation: included macros, describtions, test results**
  - https://codimd.web.cern.ch/_pflHTWdTB2TjnsWXBRtOw
- **Test can be added and executed using ctest and cmake**
- **Can be run parallel, separately, also possible to run a specific group of tests (e.g. graphs)**
- **472 tests in total**

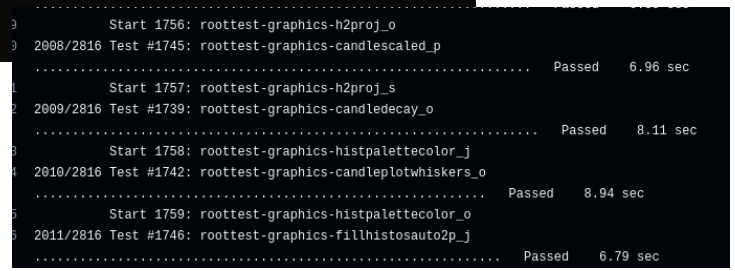
ROOT-Graphic-Test: Integration in roottest

- **Roottest:**
  - Part of ROOT's test suite
  - Tests are executed as part of ROOT's CI
- **Goal: Make tests run for all platforms of ROOT CI**
  1. Debugged more tests, excluded more parts in the comparison, disabled some tests
  2. Got rid of node.js:
     - No solution, to install necessary node modules outside of ROOT
     - Included chrome installation in ROOT CI
  3. Tried different float and double precision
  4. Script to generate new references files

ROOT-Graphic-Test: Integration in roottest

```
roottest/
├── graphics/
│   ├── json_ref/
│   ├── new_svg_ref/
│   ├── old_svg_ref/
│   ├── pdf_ref/
│   ├── macros/
│   ├── testGraphics.C
│   └── CMakeLists.txt
└── ...
```

1. Add Test with Cmake and run Ctest
2. Modified Macro (macros)
3. Test file
   1. Generate files according to test type
      - Old SVG
      - PDF
      - JSON
      - New SVG (headless-chrome-browser)
   2. Preprocess the file content
   3. Compare generated files to references
4. Output the test result



- **Headless chrome browser instead of node.js + packages**

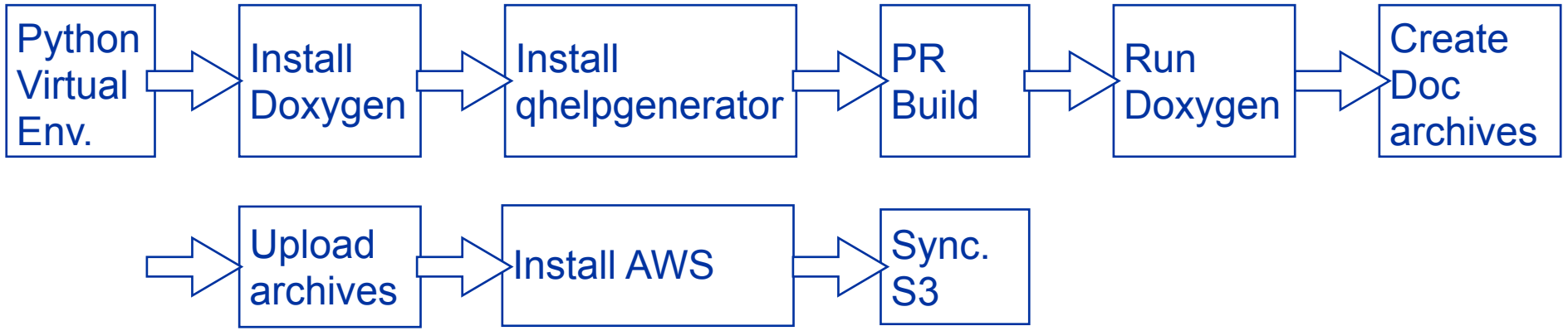
Summary

- **Developed an automated graphic testing suite**
  - 472 test on **MAC** platforms
  - Approx. 300 tests run on **all** platforms (excluding Windows)
  - Final PRs:
    - https://github.com/root-project/root/pull/16322
    - https://github.com/root-project/roottest/pull/1179
- **Open Issues:**
  - Scaleabilty of new svg graphic tests with headless web browser
  - Maintenance? **(**Manual script how to generated new reference files**)**

ROOT Doxygen Documentation

ROOT Doxygen Documenation Generation

- **Continued Jollys work**
- **Doxygen:**
  - Documentation generator tool that automatically produces software reference manuals from source code
- **Goal:**
  - Generate the ROOT Doxy Docu automatically with GitHub Actions
- root-docs.yml (main steps):



ROOT Doxygen Documenation Generation

- **Result: Documentation is uploaded as compressed artifacts on GitHub**
  - rootdoc.tar.gz
  - rootdoc.tar.xz
- **Final PR:**
  - https://github.com/root-project/root/pull/16046

- **Open Issues:**
  - The correct path to upload to S3 is missing

My Conclusion

- Reinforced graphics testing in ROOT by providing and an extensible framework
- Advanced the generation of documentation through Github Actions, by contributing to abandon the Jenkins CI
- Attended some useful talks, lectures and workshops
- Met many nice people and learnt a lot!
- Saved up some money for my master
- Started to play tennis again
- Took French lessons

Next steps?

- **Master in London at Imperial College**
  - In Computational Science and Engineering

- **THANK YOU! Especially Danilo, Serguey, Dev, Olivier and Jolly**
- **Questions?**

home.cern