Virtual-C – a Programming Environment for Learning C on Mobile Devices

D. Schmudde and D. Pawelczak
Faculty of Electrical Engineering and Computer Science
Universitaet der Bundeswehr Muenchen (UniBw M), Neubiberg, Germany

Abstract— *Virtual-C is an integrated development environment (IDE) that was primary developed for desktop systems to fit the needs of programming novices for learning C in undergraduate courses. This paper discusses the port and optimization of the IDE for the usage on mobile devices, especially on tablet computers with Android. Therefore, new fields of application were identified like an example collection for basic programming topics and an online environment for small exercises. A description language based on the JSON syntax was developed for exercise modules. Besides that, the user interface was improved by simplifying the presentation and by adding gesture support.*

Keywords: C-programming course, mobile learning, *Qt*

1. Introduction

There are different approaches of teaching a programming language in an efficient and graphic way. Mostly a suitable IDE is used to provide features for programming novices and a balanced learning curve [1]. The *Virtual-C* IDE[1] [2] was developed using the *Qt* library from *Qt Project* [3] to reach this goal and concentrates on debugging capabilities, visualizations to show the data and control flow and a mechanism for *HTML*-based exercises during practical courses.

The rising availability of mobile devices in lectures shows the need for an additional orientation of the IDE towards mobile operating systems. Therefore, more users could be reached and new methods of learning can be realized.

2. Didactics

The goal is that students can benefit from a mobile version of the IDE and are supported in their individual learning processes. The following use cases describe the main fields of application and how students can take advantage from a port in their study.

2.1 Example Collection

The integration of an example collection sorted by topics offers examples for basic programming questions that can be run and debugged to understand the codes' purpose, without the need of writing programs on the mobile device. In addition, examples can be modified directly in the editor to test and analyze alternatives.

2.2 Exercises

Programming exercises can help students to deepen their understanding of important programming concepts by leading step by step through small tasks that are collected into exercise modules (EM). After each step the achieved solution is inspected and in case of an error a small hint or a detailed feedback is given depending on the task type. The exercise templates and test files are available through a web server. In that way, the content is not static and can be customized for different courses. The progress is automatically stored on the local device so that even small time periods can be used for continuing exercises.

2.3 Editor Mode

The editor mode can be used for any C-programs or projects. Therefore, the most important developing functions are summarized into a menu to simplify the user interface (UI). It can be minimized to provide more space for coding.

3. Port and User Interface

3.1 Qt Limitations

The benefit from using *Qt* is that the application can be deployed to various platforms including mobile operating systems like *Android* or *iOS*. The drawback of this is that – until now – the support of the library for mobile systems is limited and e.g. the *WebKit* browser that is needed for the original *HTML*-based exercises is not available on mobile devices (compare [2]). For the analysis of alternative views, a description language (DL) was defined to generate *QML* representations of the EMs, see Section 4. By comparing the support on desktop and mobile systems, the presentation quality and flexibility, the handling and sustainability, *Qt Quick 2* showed the best results overall and is therefore the preferred view for exercises and additional content [4].

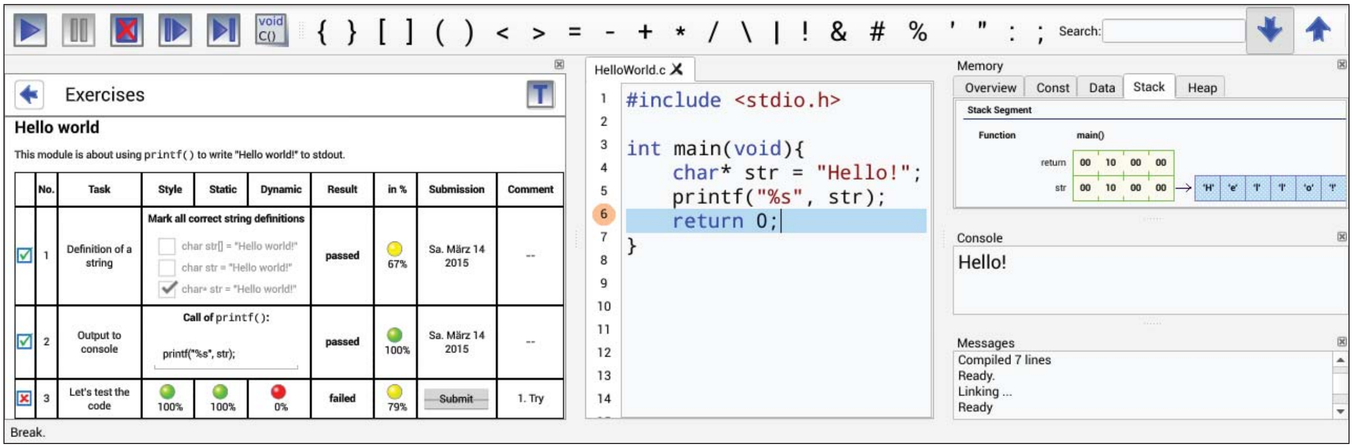
3.2 Gestures and Scalability

The UI of a mobile application has to be flexible to support a wide range of screen sizes and resolutions. This was realized by using different zoom levels and gestures that provide a familiar and easy handling of the IDE.

4. Exercise Modules

A DL was defined to combine all necessary information about an EM into a single *JSON* file.

[1] https://sites.google.com/site/virtualcide/



00 10 00 00

00 10 00 00

Hello world

This module is about using `printf()` to write "Hello world!" to stdout.

	No.	Task	Style	Static	Dynamic	Result	in %	Submission	Comment
☑	1	Definition of a string	Mark all correct string definitions □ char str[] = "Hello world!" □ char str = "Hello world!" ☑ char* str = "Hello world!"			passed	● 67%	Sa. März 14 2015	--
☑	2	Output to console	Call of `printf()`: printf("%s", str);			passed	● 100%	Sa. März 14 2015	--
☒	3	Let's test the code	● 100%	● 100%	● 0%	failed	● 79%	Submit	1. Try

```
#include <stdio.h>

int main(void){
    char* str = "Hello!";
    printf("%s", str);
    return 0;
}
```

Hello!

Compiled 7 lines
Ready.
Linking ...
Ready

Fig. 1: Screenshot of the IDE running on an *Android* device. The view on the left is used to show the content of exercise modules. It also serves as a menu to provide easy access to the main features like examples, exercises and the editor mode.

4.1 Module Description

The module description contains headings in different sizes and other text to introduce the topic and to give necessary information. The text format can be adjusted by using basic *HTML* commands.

4.2 Task Types

There are three different task types. The first type initiates a multiple choice request whereas the second type demands a textual user input. After a task submission the input is compared with the solution provided by the EM and the result is calculated as a percentage. The last type needs additional files to test the source code by compiling, linking and executing test functions for a dynamic code behavior analysis. For documentation purposes all results are added into the locally stored EMs.

4.3 Exercise Module Example

Fig. 1 above shows a visualization of an interpreted EM described in Fig. 2. It contains a header, a short introduction text and one exercise table including all three task types. (Note that in string values backslash escapes have to be used for double quotes and backslashes itself.)

5. Conclusion

The *Virtual-C IDE* focuses on debugging capabilities as well as data and control flow visualizations that are rare on mobile IDEs [4]. With additional content for new fields of application and UI enhancements, the environment is prepared for the usage on mobile devices. Yet, the original version can also take advantages from the port, because the features like examples, exercises and the gesture support can be reused on desktop systems, too. Further, the preparation of EMs or modules for the practical courses can be much faster when using the new *JSON*-based DL. Henceforth, students can practice exercises with the mobile version of the IDE at any time and any place.

```
[   { "h1": "Hello world" },
    { "text": "This module is about using <code>printf()
        </code> to write \\\"Hello world!\\\" to stdout." },
    { "exercise": [
        { "choice": {
            "task": "Definition of a string",
            "question": "Mark all correct string definitions",
            "checkbox": {
                "char* str = \\\"Hello world!\\\"": true,
                "char str[] = \\\"Hello world!\\\"": true,
                "char str = \\\"Hello world!\\\"": false
        } },
        { "scanf": {
            "task": "Output to console",
            "question": "Call of <code>printf()</code>:",
            "expected": "The complete call is needed!",
            "solution": "printf\\s*\\(\\s*\"\\s*%s\"\\s*,\\s*
                [a-zA-Z_]+[a-zA-Z0-9_]*\\s*\\)\\s*[;]?"
        } },
        { "check": {
            "task": "Let’s test the code",
            "target": "testHelloWorld.c"
        } }
] } ]
```

Fig. 2: EM example with the DL based on *JSON* syntax.

References

[1] E. Dillon, M. Anderson and M. Brown. “Comparing feature assistance between programming environments and their "effect" on novice programmers,” in Journal of Computing Sciences in Colleges, Volume 27 Issue 5, May 2012, Pages 69-77.

[2] D. Pawelczak and A. Baumann. “Virtual-C - a programming environment for teaching C in undergraduate programming courses,” IEEE Global Engineering Education Conference EDUCON, (Istanbul, Turkey, April 3-5, 2014), 1142-1148. Available: https://sites.google.com/site/virtualcide/virtualc.pdf

[3] The Qt website. [Online]. Available: http://www.qt.io/

[4] D. Schmudde, “Port and optimization of the Virtual-C IDE for the usage on mobile devices,” Bachelor thesis, UniBw M (in German: “Portierung und Optimierung der Virtual-C IDE fuer den Einsatz auf mobilen Endgeraeten”), 2014. Available: https://sites.google.com/site/virtualcide/bachelorarbeit.pdf