

TLCS: A Digital Library with Resources to Teach and Learn Computer Science

Sébastien COMBÉFIS¹, Guillaume DE MOFFARTS¹, Mile JOVANOVIĆ²

¹*Computer Science and IT in Education ASBL, Louvain-la-Neuve, Belgium*

²*Faculty of Computer Science and Engineering, Ss. Cyril and Methodius University
st. Rugjer Boshkovikj 16 Skopje, North Macedonia*

e-mail: sebastien@combefis.be, guillaume.demoffarts@csited.be, mile.jovanovic@gmail.com

Abstract. Nowadays, teaching and learning computer science is done at various ages, for several topics and for different reasons. Depending on the country, it can start from the primary school and it finishes at the higher education level, or even later if we take continuing education into consideration. Topics to be learned can be as simple as binary representation or basic programming concepts that can be taught to children to introduce them to computer science. It is also possible to teach and learn advanced data structures or algorithms optimisation, which are interesting skills for Olympiad in Informatics contestants, for example. Recently, there is a prominent number of websites and applications that have been created to help the teaching and learning of many informatics concepts. This paper presents a platform that has been designed to browse a database of resources that can be used to teach or to learn computer science. This digital library contains freely accessible resources and can be searched efficiently thanks to the proposed structure for its content. It has been designed to maximise the user's experience and to fit modern models of digital libraries. For each resource, a detailed information sheet has been produced, containing among other things pedagogical information to help teachers and learners use the resources as best as possible. This platform can also be used to train candidates to Olympiad in Informatics and other related and similar competitions.

Keywords: computer science education, digital library, pedagogical resources database, teaching and learning.

1. Introduction

Computer science is everywhere today and a lot of people are either teaching or learning some of its concepts. In some countries, computer science education starts with very young pupils still in primary school (6–12 years old), and continues with secondary school pupils (12–18 years old). Unfortunately, computer science, or just computational thinking, as a separate subject in curricula is still not widespread in the world. Hopefully, some studies on how computer science could be introduced in curricula are being

conducted (Angeli *et al.*, 2016; Barr *et al.*, 2011; Webb *et al.*, 2017). For contestants to the International Olympiad in Informatics, or other related contests, some skills of computer science are also very important, such as programming and algorithm design, for example. It is not always easy to learn computer science concepts since it often requires high level skills such as abstraction, algorithmic thinking capabilities, creative thinking, etc. Hopefully, and interestingly thanks to informatics, a lot of tools have been designed to help learning computer science concepts, and related skills.

There are a lot of tools that can be used to teach and learn computer science concepts, which are often developed as websites or applications. One main issue is that they are not always well advertised or are not so easy to find. Also, some of them being research prototypes, they often lack documentation or advices on how they can be used to support learning. The work presented in this paper tries to tackle this issue by proposing a database gathering websites and applications to teach and learn computer science.

The proposed database has been developed as a digital library, a tool that can be defined as “*a set of electronic resources and associated technical capabilities for creating, searching and using information*” (Borgman, 1999). The paper presents an online platform that has been designed as a frontend for this database, to allow teachers and learners to quickly find resources relevant to them, and to get information about how to use these resources. Since digital libraries are typically “*constructed, collected and organised by (and for) a community of users*” (Borgman, 1999), this work also proposes future extensions of the online platform to make it easier to grow and involve the community of computer science educators.

The next section presents some related work on digital libraries, their design and how to make them efficient. Section 3 then presents how the database has been structured and how the classification of the resources of the proposed digital library has been done in the frame of this work. Section 4 describes the online platform that has been developed and shows its main features. Finally, the conclusion discusses on the advantages of the proposed platform and presents the future directions that are envisioned for this digital library.

2. Related Work

Digital libraries research emerged in the early 1990s, mainly to identify how they can help and contribute to education. A digital library can be seen as a set of resources that are organised in some way to offer services to its users. Digital libraries definitely play multiple roles in teaching and learning. In particular, Marchionini *et al.* (1995) highlights three main roles: sharing resources, preserving and organising ideas and bringing together people and ideas. Moreover, digital libraries target users with different needs: formal, informal and professional learning missions. Borgman (1999) adds another dimension to the definition of digital libraries, pointing out that they can be seen as content collected on behalf of user communities for researchers and as institutions or services for librarians. More recently, Blandford (2006) focused the interaction role between users

and information that digital libraries convey, allowing for their users to find and to work with the content of the digital library.

Several digital libraries have been developed in the particular case of computer science education. Fox (1996) developed a digital library to increase the quality of learning about computer science. In the frame of his project, several changes have been made at Virginia Tech, mainly concerning the infrastructure, the pedagogy, the evaluation and the tools. The conclusion of his experiment shows that students are learning new topics in a new way, making them happy with the digital library whose number of accesses got a growth for both local and remote access. More recently, Tungare *et al.* (2007) created a syllabus repository of computer science courses across universities in the USA, with as goal to provide added value to the computer science education community. The features provided by this digital library include classifying syllabi, assisting instructors when they are creating new syllabi, and allowing the community to share their syllabi automatically and to compare syllabi for similar courses. Both these works are focused on computer science courses related content.

Today, there are a lot of online resources that can be used to learn programming and other computer science topics. At first, we may think about Open Educational Resources (OER) or Massive Open Online Courses (MOOC) that were made possible thanks to the tremendous growth of ICT in recent years, opening up new opportunities for education, and accessible ways to enjoy quality teaching and learning at all levels (Jemni *et al.*, 2017). In addition to these resources, most of the time associated to courses, people can also learn a lot through programming contests, such as Olympiads in Informatics and other related programming competitions (Combéfis *et al.*, 2014), or with games (Combéfis *et al.*, 2016). The approaches presented in the two latter papers follow the current trend of new models of open and distributed learning (Downes, 2017). As summarised by the author, the important changes are the fact that the learner must go from passive to active and from formal to informal, which is possible thanks to open and distributed learning.

Distance-based education fostered the development of educational tools that can be used online to support teaching and learning. Some of these tools have been developed for MOOCs, such as code executors and graders (Combéfis *et al.*, 2015; Bey *et al.*, 2018) and graph sketchers (French *et al.*, 2017), for example. Other tools are stand-alone applications that can be used independently, online or just locally after installation. All these tools have been developed thanks to computer science, and improve the learners' experience.

For the particular case of computer science education, there are also a lot of tools and prototypes that have been thought about and developed by researchers. For example, Combéfis *et al.* (2013) presented a tool with interactive problems that can guide learners from the understanding of the problem to the coding of a solution for it. Another example comes from Guo (2013) who developed a tool to visualise the execution of any program for learners to map static textual representation (source code) to what is dynamically happening in the computer (execution). A last example, designed by Folland (2016), is a tool to visualise the execution of SQL INSERT statements to highlight how their results are built from source tables.

As mentioned above, another useful resource to learn programming, and other computer science topics, is games (Combéfis *et al.*, 2016). They are playing a large role in teaching computing in higher education, as testified by some reviews. For example, Batistella *et al.* (2016) pointed out that several computing knowledge areas are covered by games, software engineering and programming fundamentals being the most common covered fields. Nevertheless, games are not a panacea as highlighted by Rondon *et al.* (2013), whose study showed that compared to traditional learning, games are only interesting for short-term knowledge retention, at least for medical education. At least, games help to get learners involved with the learning activities, as reported by Schmitz *et al.* (2011), following their experiment.

The examples of tools just presented show that more and more resources are being designed to help the learning of several topics in computer science, namely programming, database, algorithm thinking, etc. Nevertheless, it is not always easy to find such resources. Grissom *et al.* (1998) highlighted the need for a digital library of computer science teaching resources, years ago. More recently, Dichev *et al.* (2012) explained that looking for an appropriate resource is a frequent activity in the job of teaching. Whereas digital libraries of OERs do exist, in particular in the context of courses, no such digital library seems to exist for more general tool resources that cover various computer science topics.

To be efficient, usable and useful, a digital library must be well designed, in particular in the frame of education. Sumner *et al.* (2003) highlighted several key factors that influence the perception of educators about the quality of digital libraries, when used for education. The main results show that a good digital library should favour resources (1) that encourage active learning, (2) that do not result in any bias regarding political or commercial orientations, (3) that limit the access to resources with advertising, (4) that are usable and well-designed to ease the navigation and usability, (5) and that avoid any distractions affecting the attention of learners. These observations have been pointed out by learners as well as by teachers.

Other studies have been made about the interface of digital libraries. In particular, Druin *et al.* (2001) put a focus on this need, especially for children that do not want to just search for information, but also need to use it and need a reason to browse for an item. Compared to a traditional library, a digital library may lack social interaction. Ackerman (1994) insists that social exchanges and interaction are important. The design and use of a digital library should not be limited to the technical mechanisms and the access of information. Mechanisms to make these social interactions possible and to foster them should therefore be thought about. Gazan (2018) goes one step further to include content creators, in addition to content consumers, as an important set of users of digital libraries. The author highlighted the possibility to include user-generated content into digital collection items, therefore increasing the social interactions. Finally, Sumner *et al.* (2004) analysed three models that can be used as approaches to educational digital library design. Their conclusion is that digital libraries can be used (1) as cognitive tools to support learning and help users to catch the sense of the activities, (2) as component repositories to focus on how resources from the digital library

are produced and distributed, (3) and as knowledge networks to foster social interactions and knowledge building and sharing.

The different elements highlighted by these related works have all been somewhat taken into account for the design of the platform presented in this paper.

3. Classification of Websites and Applications

The websites and applications that can be used to teach or to learn computer science are classified according to several criteria. The proposed characterisation is meant to help teachers and learners to choose the most suited and relevant website or application that fits their needs. It should also help teachers to use the resources in the most effective way. For the platform to be powerful, yet flexible, and to ease the development of a community of users around the platform resources, a detailed characterisation of the resources is proposed in this paper and explained in this section.

3.1. Category


The first classification criterion is related to the kind of service that the website or the application is providing. According to the resources that have been considered and analysed in the frame of this work, six main categories have been identified:

- Directory.
- Visualiser.
- Animated tutorial.
- Playground.
- Interactive tutorial.
- Game.

3.1.1. Directory

The *directory* category gathers resources that allow their users to navigate through a collection of resources, technologies, tools, etc. Websites or applications from this category help learners to discover resources related to the same topic or field of study. For example, the “*NoSQL Databases*” website, shown on Fig. 1, maintains a large list of NoSQL databases engines organised according to their main paradigm. It is an interesting resource for anyone who discovered the NoSQL world and wants to explore the existing engines or to choose one for a project.

Resources from this category are similar to the “awesome list” movement whose main goal is to gather a curation of awesome stuff in lists (Sorhus, 2019). The main difference between a simple collection or aggregation and a curation is that the latter involves a selection of content based on quality (Dale, 2014). For example, Caero-Rodríguez *et al.* (2013) proposed a social curation platform for OERs.



Your Ultimate Guide to the
Non-Relational Universe!

[including a historic [Archive](#) 2009-2011]
News Feed covering some changes [here](#) !

NOSQL DEFINITION:Next Generation Databases mostly addressing some of the points: being non-relational, distributed, open-source and horizontally scalable.

The original intention has been modern web-scale databases. The movement began early 2009 and is growing rapidly. Often more characteristics apply such as: schema-free, easy replication support, simple API, eventually consistent / BASE (not ACID), a huge amount of data and more. So the misleading term "nosql" (the community now translates it mostly with "not only sql") should be seen as an alias to something like the definition above. (based on 7 sources, 15 constructive feedback emails (thanks!) and 1 disliking comment. Agree / Disagree? [Tell me so!](#) By the way: this is a strong definition and it is out there here since 2009!)

LIST OF NOSQL DATABASES [currently >225]

Core NOSQL Systems: [Mostly originated out of a Web 2.0 need]

Wide Column Store / Column Families

[Hadoop / HBase](#) API: Java / any writer, Protocol: any write call, Query Method: MapReduce Java / any exec, Replication: HDFS Replication, Written in: Java, Concurrency: ?, Misc: Links: 3 Books [[1](#), [2](#), [3](#)], [Guru99 Article](#) >>

[MapR](#), [Hortonworks](#), [Cloudera](#) Hadoop Distributions and professional services .

[Cassandra](#) massively scalable, partitioned row store, masterless architecture, linear scale performance, no single points of failure, read/write support across multiple data centers & cloud availability zones. API / Query Method: CQL and Thrift, replication: peer-to-peer, written in: Java, Concurrency: tunable consistency, Misc: built-in data compression, MapReduce support, primary/secondary indexes, security features. Links: [Documentation](#), [PlanetC*](#), [Company](#).



[Scylla](#) Cassandra-compatible column store, with consistent low latency and more transactions per second. Designed with a thread-per-core model to maximize performance on modern multicore

NoSQL RELATED EVENTS:

- June 26-27 2018 MongoDB World [»](#)

Register your event 4free: [»](#)

NoSQL ARCHIVE

the multi-model NoSQL DB

NoSQL FORUMS

- Global NoSQL Forum [»](#)
- Forum Berlin [»](#)
- Forum France [»](#)
- Forum Japan [»](#)

NoSQL NEWS FEEDS

- MyNoSQL by Alex P [»](#)
- On Twitter: nosqlupdate [»](#)
- NoSQL Weekly [»](#) * new *
- HighScalability Blog [»](#)

Fig. 1. The *NoSQL Databases* website maintains a collection of NoSQL database engines that are organised according to their main paradigm.

3.1.2. Visualisation

To teach and to learn new concepts, it can help a lot to be able to visualise, in some way, the new concepts. In particular, people who are more sensitive to visual modalities will benefit from such visualisations. The second and third categories contain resources that propose tools to visualise concepts.

The *visualiser* category contains tools that can produce visualisations, either static or dynamic ones. The goal of these visual elements is to help you to represent yourself the concepts you are supposed to learn. Such tools can also be used for teaching purpose, to provide visual examples to your students (Fouh *et al.*, 2012). As highlighted by Naps *et al.* (2002), visualisation is only effective if it engages learners in an active learning activity. It is therefore important to provide explanations on how the use the visualisation tool to support learning.

For example, the “*viSQLizer*” platform (Folland, 2016), shown on Fig. 2, is a prototype visual learning tool for SQL. The tool builds and shows animations to illustrate how the result of a SELECT query is built by extracting rows from the involved tables. This can be used in an introductory course on databases and queries, to illustrate how data

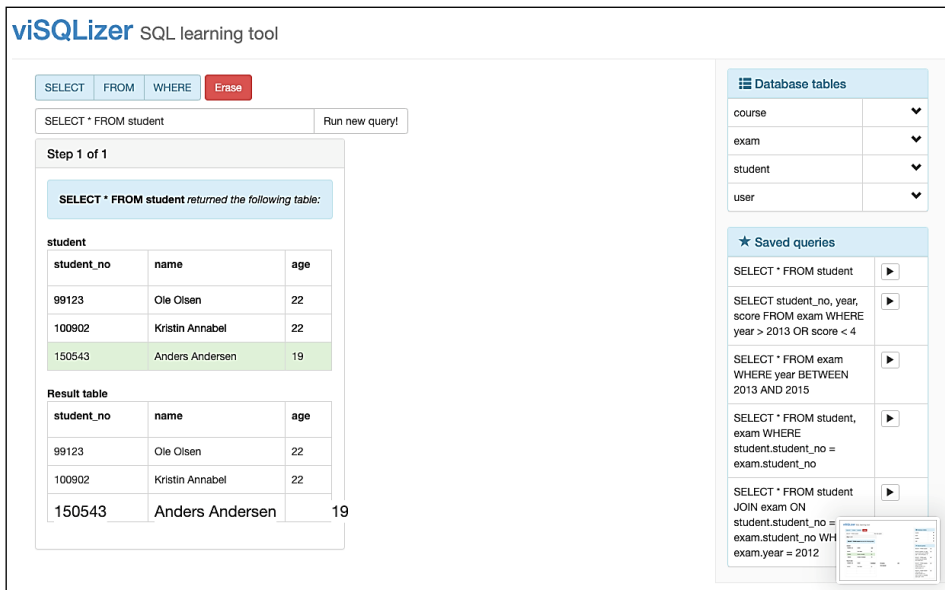


Fig. 2. The *viSQLizer* prototype visual learning tool helps learners to understand how the result of a SELECT query written in the SQL language builds its result from tables.

organised in tables can be scanned through to get the result of a given request. It is also interesting to see the different steps behind a SELECT query, starting with row filtering followed by column projection.

Resources from this category, if correctly used, will engage their users in their own learning. This exactly matches good visualisation tools as defined by Naps *et al.* (2002). Unfortunately, most of them being the result of PhD or master thesis, they lack pedagogical documentation on how to use them effectively. Also, they are not accompanying the learner within a learning path.

The *animated tutorials* category is dedicated to websites and applications that provide a tutorial meant to teach new concepts, by presenting you direct examples with the produced results (Rodger, 2002). It goes one step further compared to the visualiser category, in the sense that the visualisation are embedded within a tutorial that guides you for your learning. For example, the “*Unfolding the Box Model*” website, shown on Fig. 3, shows you how do CSS 3D transforms work. Each page of the tutorial just shows you directly the result of the transforms that are presented.

Visualisation tools allow the user to ask for a visual representation of a given input, such as an SQL query, an operation on a given data structure, an execution of an algorithm for a problem instance, etc. Once the input has been provided, the tool shows a visualisation that the user is just watching. In the case of animated tutorials, the user is presented a sequence of visual animations to gradually explain the user concepts, like a tutorial would have done.

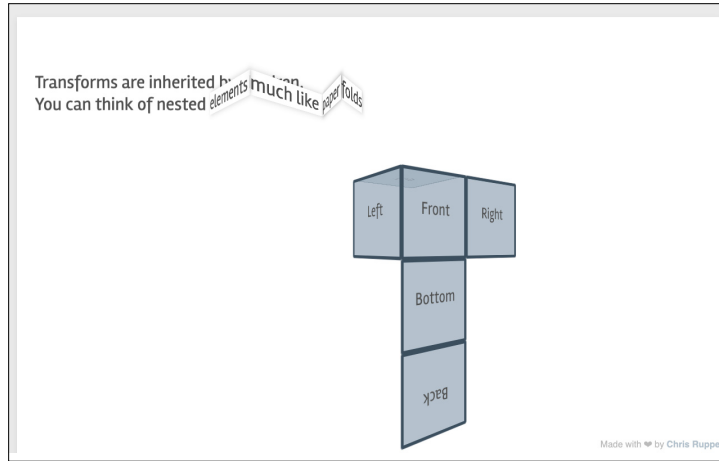


Fig. 3. The *Unfolding the Box Model* website shows you how CSS 3D transforms work with concrete examples that have been put as a single animated tutorial.

3.1.3. Interaction

The next two categories add the ability for the user to interact actively with the website or the application. The user is offered the possibility to play with his/her own examples and gets a direct feedback. In this way, the user can be challenged and put in the centre of his/her own learning, which will contribute to improve what he/she will learn and assimilate (Bork, 2001).

The *playground* category is for websites and applications where the user can enter codes, problem instances, situation descriptions, etc. and execute them to visualise and get the result directly. Such resources are useful for the user to be able to play without the need to install anything on his/her computer. For example, the “*RxViz*” website, shown on Fig. 4, allows you to play with RxJs observables in an animated playground. It makes it possible for you to write your own code, or even to take one of the proposed examples, to execute it and to get the result in a visual way.

Playgrounds are very similar to simple visualisation tool, except that they provide more freedom and can visualise much more complex objects, especially code. These tools are showing a visual execution of the code along with the execution of the latter. Changing the code and executing it again will directly update the visualisation. Sometimes, it is also possible to directly interact with the visualisation, and the code could be updated accordingly.

The *interactive tutorial* category is one step further the animated tutorial, in the sense that the user will be challenged and asked to interact with the animations. As a tutorial, it is accompanying the learner during the learning process. And as an interactive tutorial, it asks the learner to take part to the learning process through different kinds of interactions. For example, the “*Computer Science Field Guide*” is an online interactive book that can be used to teach various computer science concepts to high school students. It provides interactive exercises throughout the book that allow the learners to experiment what they learned. Fig. 5 shows one of the interactive exercises that are proposed on the website.

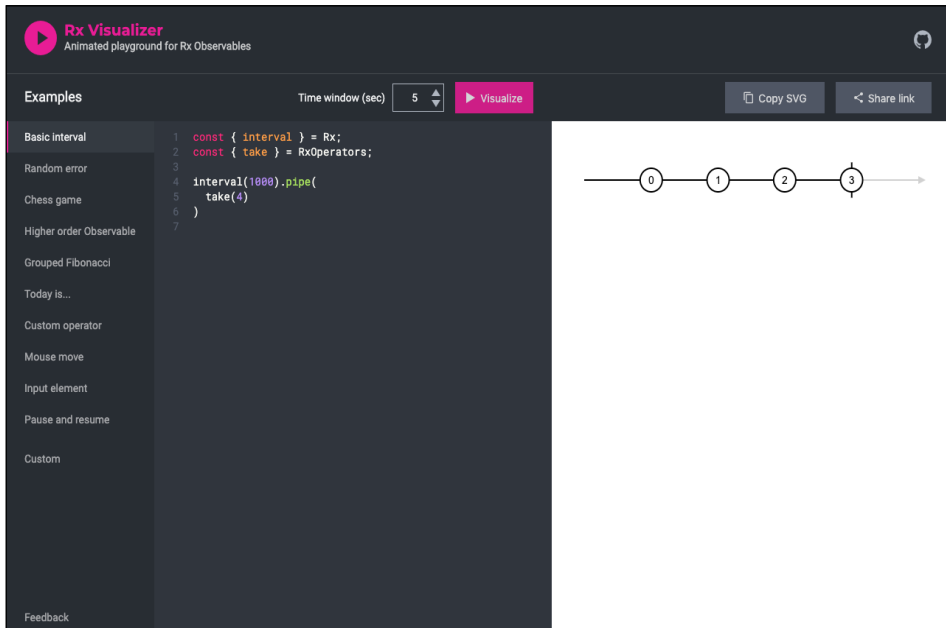


Fig. 4. The *RxViz* website is a playground where you can write and execute programs using RxJS observables and get a visual interpretation of the result.

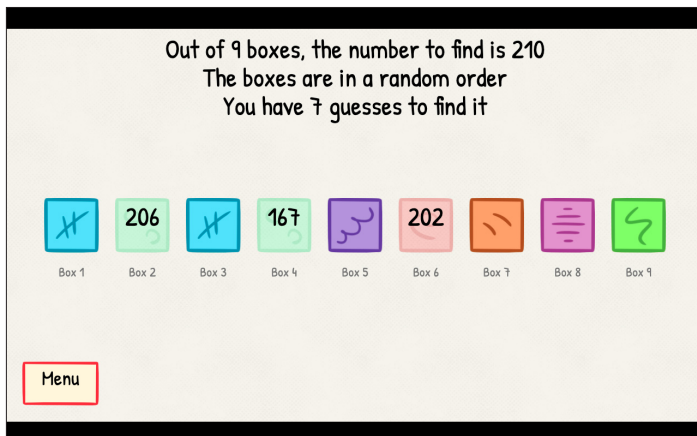


Fig. 5. The *Computer Science Field Guide* is a website that proposes interactive exercises, as part of an online book, to help its learners to understand the new concepts.

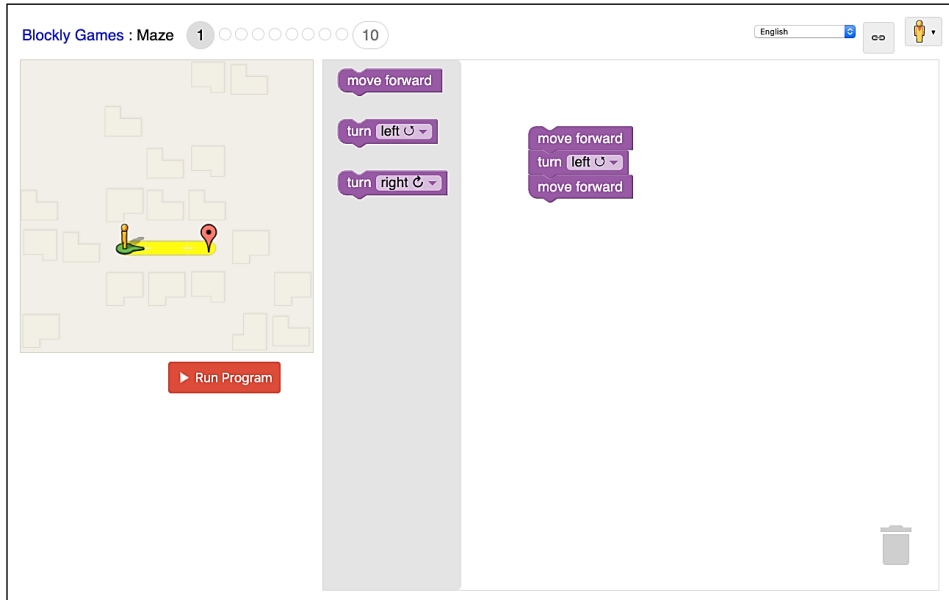


Fig. 6. The *Blockly Games* website proposes a game with several tasks that have to be solved using a blocs-based visual programming language in order to win the game.

3.1.4. Game

Finally, websites and applications from the last category, namely *game*, provide the most interactive experience to the learners and require them the largest involvement. It also tries to motivate them with the addition of goals, scoreboards, competition, etc., compared to the interactive tutorials (Combéfis *et al.*, 2016).

For example, the “*Blockly Games*” website challenges its users by asking them to solve several tasks whose solutions are programs written with a block-based visual programming language, similar to Scratch. Each task can be seen as a small game, each of these being one level in a bigger game. One of the tasks being solved is shown on Fig. 6.

3.1.5. Overlapping Categories

The six categories presented in this section are overlapping, meaning that some websites or applications can belong to more than one category. For example, the “*RxViz*” website is at the same time a playground and a visualisation tool since it allows you to write and execute any code but presents visually the result of the execution. Another example is the “*SQL Island*” website, presented in section 4, which is at the same time a game and an interactive tutorial. The platform described in this paper allows you to easily navigate the resources database according to the categories.

3.2. Language and Field

The two others classification criteria are the programming *languages* and the computer science *fields* that are covered by the website or application. The first criterion is optional and the second one is mandatory. A computer science resource is indeed always related to at least one field but does not always concern a programming language. The presented platform makes it possible to search for teaching and learning resources based on these two criteria.

The possible values for the programming language criterion are quite clear but it is less obvious for the computer science field. In this work, general fields such as database, programming, data structure, artificial intelligence, etc. have been used. Another possibility would be to use the *ACM Computing Classification System (CCS)*, but it may be too complex for the targeted users for the proposed platform.

Classification by categories, programming language and computer science fields can help to better identify and attract people from existing communities of interest. For example, people interested in resources related to the Python programming language, for machine learning, could easily find them with the proposed platform.

3.3. Level

Teaching and learning computer science is done at various ages and level of education. For each website and application available in the platform described in this paper, the most suited age *levels* are indicated. Five levels have been identified:

- Children goes until 12 years old, that is the end of primary school.
- Junior goes from 12 to 15, that is, lower secondary school.
- Senior goes from 15 to 18, that is, higher secondary school.
- BSc is for bachelor students.
- MSc is for master students.

The identified levels are indicative and correspond to the most suited age groups with which the resource can be used to teach or to learn the concepts conveyed by the resource. This way to organise the resource is directly related with the community of teachers. For example, a primary school teacher will indeed first search for resources relevant for the age of his/her pupils.

3.4. Pedagogical Information

Finally, to help teachers and learners to use the resources available on the proposed platform, various *pedagogical information* can be added to each website or application. Their purpose is to propose a guide to use the resource as best as possible. Three kinds of information can be provided: prerequisite, learning outcomes and methodology.

The *prerequisites* summarise what knowledge should be mastered to be able to use the resource to learn the conveyed concepts. The content of this section should be written according to the proposed age levels. The *learning outcomes* list what the student will be able to do after he/she used the resource in the frame of a learning activity. This section can also contain information about the content proposed by the resource. Finally, the *methodology* explains how the resource can be used or how it is supposed to be used by its original creators and designers.

The three kinds of information are of course not relevant for all the resource categories. For example, pure playgrounds will typically lack learning outcomes and methodologies. Also, learning outcomes could depend on how the resource is used by a teacher in an activity. On the platform presented in this paper, the proposed pedagogical information is written to be consistent, meaning that the learning outcomes are to be read with the proposed methodology on how to use the resource.

4. Interactive Platform

This paper proposes an online platform, called TLCS for “*Teaching and Learning Computer Science*”, available online at the following address: <https://tlcs.csited.be>. It is only available in English for now but is ready for internationalisation. Fig. 7 shows the page describing the “*SQL Island*” website, a game to learn the fundamentals of the SQL database querying language (Schildgen, 2014).

4.1. Structure of the Platform

The layout of the page is structured in three columns. A navigation tool is available on the left part to allow the user to browse the resources by categories, programming languages, computer science fields or levels of education. It is also possible to make some cross-searches by clicking on the magnifying glass and selecting the tags you are interested in. For example, you could search for resources that are interactive tutorials in the form of games, such as illustrated on Fig. 8.

An information panel is visible on the right part to show all the categories, programming languages and computer science fields of the resource. You can also directly see the levels of education and access the website of the resource through this information panel. Depending on the resource, some of the information may or may not be available. Fig. 9 shows the information available for “*SQL Island*”.

Finally, the central column shows a short description with screenshots directly followed by the pedagogical information. Two last optional sections can be available, depending on the resource. The *service* section describes the kinds of service provided by the resource, such as cooperative game, API, possibility to save or share, etc. The *references* section provides scientific references to papers presenting the website or application, when available.

TLCS v1.0
en ⌵ About

Category Language Field Q

- Interactive tutorial
- CSS Diner
- SQL Island
- Grid Garden
- Game
- Playground
- Animated tutorial

SQL Island

SQL Island is an **adventure game** where the hero is stuck on an island and tries to escape from it. In order to communicate with the inhabitants of the island, you have to speak the **SQL database querying language**, the only language they can understand. Playing the game will teach you the fundamentals of SQL in an entertaining and funny way.



It is possible to switch the language to English through the third menu item of the options menu.

Information

- Categories
 - Game, Interactive tutorial
- Fields
 - Database, Programming
- Levels
 - Senior, BSC
- Languages
 - SQL
- Website
 - SQL Island

Prerequisites

To be able to play this game, you just need to understand how data can be organised in a tabular way, such as in a spreadsheet, for example.

Learning outcomes

After you escaped from the island, that is, you successfully finished the game, you will be able to understand the **fundamentals of the SQL** database querying language. More precisely, you will discover:

- *SELECT statements* to extract rows from a table;
- *WHERE clauses* to restrict the rows affected by queries on a table and *AND, OR and LIKE operators* to build complex conditions;
- *INSERT INTO statements* to insert rows in a table;
- *NULL values* to indicate missing values for some columns in a row;
- *UPDATE statements* to update existing rows from a table;
- *ORDER BY clauses* to order the extracted rows in ascending or descending order;
- *Natural join operations* to extract rows by combining rows extracted from several tables;
- *COUNT function* to count the number of rows, *SUM function* to sum the values of a column and *AVG function* to compute the average value of a column, for all the rows satisfying the criteria of the WHERE clause;
- *GROUP BY clause* to group rows having the same value for some columns together, optionally used with aggregate functions such as COUNT, SUM, etc.

Methodology

During the game, you will first have to **understand queries** that you want to perform on the villages of the island, their inhabitants and the items they have. After that, you will have to **write them using SQL**. To help you, the system will show you some queries with the results they produced, from time to time.

The game is **incremental** and each step teaches you a new construct. Moreover, **some hints** about which construct you should use for the query you have to perform are sometimes provided. Finally, you can **try any SQL query** at any time and see the produced result, with a comment to help you if you have the wrong answer.

Services

The website provides you a game that you have to **play in one go**. You can also restart the game at any time, if you deleted some rows by mistake, for example.

References

- Schildgen J., & DeBloch S. (2015). SQL-Grundlagen spielend lernen mit dem Text-Adventure SQL Island. In Seidl, T., Ritter, N., Schöning, H., Sattler, K.-U., Härdter, T., Friedrich, S. & Wingerath, W. (Eds.) *Datenbanksysteme für Business, Technologie und Web (BTW 2015)* (pp. 687-690). Bonn: Gesellschaft für Informatik e.V.
- Schildgen J. (2014). SQL Island: An Adventure Game to Learn the Database Language SQL. In *Proceedings of the 8th European Conference on Games Based Learning (ECGBL 2014)* (pp. 137-138).
- Schildgen J., & DeBloch S. (2013). „Gib mir so viel Gold, wie die Metzger im Nachbardorf zusammen besitzen und ich lasse den Piloten frei!“ – Spielbasiertes Lernen von SQL-Grundlagen. *Datenbank Spektrum*, 13(3), 243-249.

Fig. 7. Each website or application is described with a complete information sheet that contains categorisation and pedagogical information.

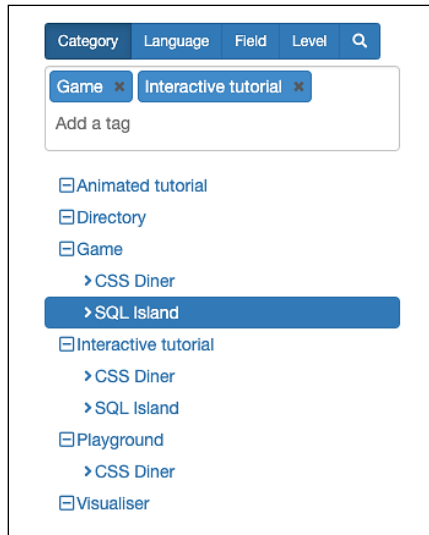


Fig. 8. The left part of the TLCS platform allows you to browse all the resources by categories, programming languages, computer science fields or levels of education. You can also search for resources by tags.

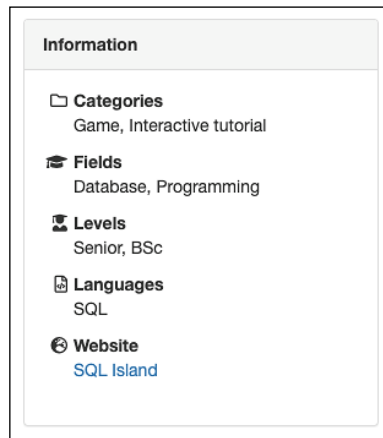


Fig. 9. The right part of the TLCS platform shows you all the available information on the resource you are looking at, allowing you to quickly characterise the resource.

The online platform has been designed to be simple to use and so that the information is clearly presented. It is also very light and runs in any modern browser since it relies on the recent versions of JavaScript. It has been developed with Angular.js for the frontend part and with Bootstrap 3 for the style. The database is just a simple JSON file, in fact one for each language (only English and French being available for now).

4.2. Database Population

The proposed digital library is meant to be populated by the community. It is indeed a way to ensure coherence between the needs of the community and the content offered on the platform. For a digital library to be good, and to ensure that its content is of high quality, some review and/or control mechanism must be put in place. Entries of the database for each resource must be correct, with exact, complete, relevant and up-to-date information.

To achieve the latter requirements, the proposed platform has a public page that anyone can use to propose a new resource for addition. Each proposition has to be reviewed, and is possibly corrected, before being accepted. The *Computer Science and IT in Education* non-profit organisation is currently in charge of this control and acceptance process. It may be opened to new partners in the future, especially when content in other languages than English will be made available. This way to proceed ensures good quality content while keeping the platform somewhat open to the community and its users.

4.3. Social and Community Aspects

As presented in the related work section, social aspects are very important for a digital library to be useful, used and for it to support learning and also knowledge sharing. Some elements to foster social interactions and to highlight community aspects have been thought about for the proposed platform, even if not implemented in the current version yet.

Users will be able to have an account on the platform and decide of their own tags for the resources. This feature allows them to organise the resources with their own categorisation. The second feature is the ability for the users of the platform to grade each resource with stars, so that the best resources will get more stars than the less good resources.

5. Conclusion

To conclude, this paper presents a digital library with websites and applications that can be used to learn computer science concepts. This database is structured so that to be easily queried to find useful and relevant resources to teach or to learn new concepts. Its design and the way its information is structured has been thought about regarding advices about how to make an efficient digital library.

The paper proposes a multi-criteria categorisation of all the resources contained in the database. To help people to search through the database, an online platform has been developed and made available to the community. It proposes a simple yet powerful and

ergonomic interface to look at the resources from the database. This interface brings several intuitive and useful ways to extract relevant information from the developed database. It has still to be improved, in particular to take into account the content creators and to include user-generated content.

Compared to other kinds of digital library that exists, which are mostly focused on OERs for teachers, and in particular for higher education, this work proposes a database that can also be used by learners, from the youngest ones to adults who already graduated. The proposed digital library, which already contains about twenty resources, supports the creation, the search and the use of resources. The information that is provided with each resources supports learning, in the most efficient way as possible.

Future work includes the translation of the platform in several languages as well as the translation of the database content, to widen the community that could therefore enjoy the available data. New resources will also be added, especially applications that can be used on smartphones. Finally, the platform and its interface will also be improved, with the possibility to add comments and notes for each resource, for example. Last but not least, surveys must be conducted with teachers, to evaluate whether the proposed platform fits their needs.

References

- Ackerman, M.S. (1994). Providing Social Interaction in the Digital Library. In: *Proceedings of the 1st Annual Conference on the Theory and Practice of Digital Libraries*. 198–200.
- Angeli, C., Voogt, J., Fluck, A., Webb, M., Cox, M., Malyn-Smith, J., Zagami, J. (2016). A K-6 Computational thinking curriculum framework: Implications for teacher knowledge. *Educational Technology & Society*, 19(3), 47–57.
- Barr, V., Stephenson, C. (2011). Bringing computational thinking to K-12: What is involved and What is the role of the computer science education community? *ACM Inroads*, 2(1), 48–54.
- Batistella, P., von Wangenheim, C.G. (2016). *Games for Teaching Computing in Higher Education – A Systematic Review*.
- Bey, A., Jermann, P., Dillenbourg, P. (2018). A comparison between two automatic assessment approaches for programming: An empirical study on MOOCs. *Educational Technology & Society*, 21(2), 259–272.
- Blandford, A. (2006). Interacting with information resources: digital libraries for education. *International Journal of Learning Technologies*, 2(2/3), 185–202.
- Borgman, C. (1999). What are digital libraries? Competing visions. *Information Processing and Management*, 35(3), 227–243.
- Bork, A. (2001). Tutorial learning for the new century. *Journal of Science Education and Technology*, 10(1), 55–71.
- Caeiro-Rodríguez, M., Pérez-Rodríguez, R., García-Alonso, J., Manso-Vázquez, M., Llamas-Nistal, M. (2013). AREA: A social curation platform for open educational resources and lesson plans. In: *Proceedings of the 2013 IEEE Frontiers in Education Conference (FIE)*. 795–801.
- Combéfis, S., Van den Schrieck, V., Nootens, A. (2013). Growing algorithmic thinking through interactive problems to encourage learning programming. *Olympiads in Informatics*, 7, 3–13.
- Combéfis, S., Wautelet, J. (2014). Programming trainings and informatics teaching through online contest. *Olympiads in Informatics*, 8, 21–34.
- Combéfis, S., Paques, A. (2015). Pythia reloaded: An intelligent unit testing-based code grader for education. In: *Proceedings of the 1st International Workshop on Code Hunt Workshop on Educational Software Engineering (CHESE 2015)*. 5–8.
- Combéfis, S., Beresnevičius, G., Dagienė, V. (2016). learning programming through games and contests: Overview, characterisation and discussion. *Olympiads in Informatics*, 10, 39–60.
- Dale, S. (2014). Content curation: The future of relevance. *Business Information Review*, 31(4), 199–205.

- Dichev C., Dicheva, D. (2012). Open Educational Resources in Computer Science Teaching. In: *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education (SIGCSE 2012)*. 619–624.
- Downes, S. (2017). New Models of Open and Distributed Learning. In: *Open Education: from OERs to MOOCs*. Berlin/Heidelberg: Springer-Verlag, 1–22.
- Druin, A., Bederson, B.B., Hourcade, J.P., Sherman, L., Revelle, G., Platner, M., Weng, S. (2001). In: *Proceedings of the 1st ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL 2001)*. 398–405.
- Folland, K.A.T. (2016). viSQLizer: Using visualization for learning SQL. In: *Proceedings of the 29th Norsk Informatikkonferanse (NIK 2016)*.
- Fouh, E., Akbar, M., Shaffer, C.A. (2012). The role of visualization in computer science education. *Interdisciplinary Journal of Practice, Theory, and Applied Research*, 29, 95–117.
- Fox, E.A. (1996). Interactive learning with a digital library in computer science. In: *Proceedings of 26th Frontiers in Education Annual Conference (FIE 1996)*. 415–419.
- French, J., Segado, M.A., Ai, P.Z. (2017). Sketching graphs in a calculus MOOC: Preliminary results. In: *Frontiers in Pen and Touch*. Cham: Springer, 93–102.
- Gazan, R. (2008). Social annotations in digital libraries collections. *D-Lib*, 14, 11/12.
- Grissom, S., Knox, D. Copperman, E., Dann, W., Goldweber, M., Hartman, J., Kuittinen, M., Mutchler, D., Parlante, N. (1998). Developing a digital library of computer science teaching resources. In: *Working Group Reports of the 3rd Annual SIGCSE/SIGCUE ITiCSE Conference on Integrating Technology into Computer Science Education (ITiCSE-WGR 1998)*. 1–13.
- Guo, P.J. (2013). Online Python tutor: Embeddable Web-based program visualization for CS education. In: *Proceedings of the 44th ACM Technical Symposium on Computer Science Education (SIGCSE 2013)*. 579–584.
- Jemni, M., Kinshuk, Khribi, M.K. (2017). *Open Education: From OERs to MOOCs*. Berlin/Heidelberg: Springer-Verlag.
- Marchionini, G., Maurer, H. (1995). The roles of digital libraries in teaching and learning. *Communication of the ACM*, 38(4), 67–75.
- Naps, T.L., Rößling, G., Almstrum, V., Dann, W., Fleischer, R., Hundhausen, C., Korhonen, A., Malmi, L., McNally, M., Rodger, S., Velázquez-Iturbide, J.A. (2002). Exploring the role of visualization and engagement in computer science education. In: *Proceedings of Working Group Reports from ITiCSE on Innovation and Technology in Computer Science Education (ITiCSE-WGR 2002)*. 131–152.
- Rodger, S.H. (2002). Introducing computer science through animation and virtual worlds. In: *Proceedings of the 33rd SIGCSE Technical Symposium on Computer Science Education (SIGCSE 2002)*. 186–190.
- Rondon S., Sassi, F.C., Furquim de Andrade C.R. (2013). Computer game-based and traditional learning method: A comparison regarding students’ knowledge retention. *BMC Medical Education*, 13, 30.
- Schildgen J. (2014). SQL island: An adventure game to learn the database language SQL. In: *Proceedings of the 8th European Conference on Games Based Learning (ECGBL 2014)*. 137–138.
- Schmitz B., Czauderna, A., Klemke, R., Specht, M. (2011). Game based learning for computer science education. In: *Proceedings of the Computer Science Education Research Conference (CSERC 2011)*. 81–86.
- Sorhus, S. (2019). *Awesome*. <https://github.com/sindresorhus/awesome>
- Sumner, K., Khoo, M., Recker, M., Marlino, M. (2003). Understanding educator perceptions of “Quality” in digital libraries. In: *Proceedings of the 3rd ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL 2003)*. 269–279.
- Sumner, T., Marlino, M. (2004). Digital libraries and educational practice: A case for new models. In: *Proceedings of the 4th ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL 2004)*. 170–178.
- Tungare, M., Yu, X., Cameron, W., Teng, G., Pérez-Quinones, M.A., Cassel, L., Fan, W., Fox, E. (2007). Towards a syllabus repository for computer science courses. In: *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education*. 55–59.
- Webb, M., Davis, N., Bell, T., Katz, Y.J., Reynolds, N., Chambers, D.P., Syslo, M.M. (2017). *Education and Information Technologies*, 22(2), 445–468.



S. Combéfis obtained his PhD in engineering in November 2013 from the Université catholique de Louvain (UCLouvain). He is currently working as a lecturer at the ECAM Brussels Engineering School, where his courses focus on computer science. He also obtained an advanced master in pedagogy in higher education in June 2014. Co-founder of the Belgian Olympiad in Informatics (be-OI) in 2010, he later introduced the Bebras contest in Belgium in 2012 and at the same time founded CSITEd. This non-profit organisation aims at promoting computer science in secondary schools.



G. de Moffarts is a master student in computer science at Université catholique de Louvain (UCLouvain). He is interested in computer science and electronics, and very curious about engineering and new technologies, such as 3D printing, artificial intelligence and the internet of things. He is also involved in the CSITEd non-profit organisation, taking part on several projects it organises. He was also recently the deputy leader of a Belgian delegation to the IBU Olympiad in Informatics 2019 that was held in Skopje, North Macedonia.



M. Jovanov is an associate professor at the Faculty of Computer Science and Engineering, Ss. Cyril and Methodius University, in Skopje. As the President of the Computer Society of Macedonia, he has actively participated in the organization and realization of the Macedonian national competitions and Olympiads in informatics since 2001. He has been a team leader for the Macedonian team at International Olympiads in Informatics since 2006. His research interests include development of new algorithms, future web, and e-education.