

---

# Multi-Game Code-Duel for Learning Programming Languages

Sven Groppe, Ian Pösse

Institute of Information Systems (IFIS), University of Lübeck, Ratzeburger Allee 160, D-23562 Lübeck, Germany,  
[groppe@ifis.uni-luebeck.de](mailto:groppe@ifis.uni-luebeck.de), [ian.poesse@student.uni-luebeck.de](mailto:ian.poesse@student.uni-luebeck.de)

---

## ABSTRACT

Software developers compose computer instructions following the rules defined in programming languages for the purpose of automatic information processing. However, different programming languages have different syntax and semantic rules, and support different programming paradigms and design patterns. Learning a programming language needs many efforts and much practicing in order to master the rules and apply the patterns. Learning multiple programming languages at the same time, of course, needs more efforts. In this work we develop the concept of multi-game and an e-learning platform called “Multi-Game Platform for Code-Duels” for learning multiple programming languages easily and efficiently. A multi-game is a video game, which consists of several mini-games. Dividing a big game into mini-games reduces the development efforts and implementation complexity. “Builders” is a multi-game developed in our platform consisting of three mini-games. Each mini-game can be solved by implementing a program by learners using different languages. Using our multi-game platform, each mini-game of Builders can be developed easily and played independently of the other mini-games. Finally, a user evaluation over our multi-game platform is performed, where users rate our multi-game approach and platform for learning programming languages very positively.

## TYPE OF PAPER AND KEYWORDS

Regular research paper: *e-learning, code-duel, multi-game, learning programming, script and query languages, learning programming frameworks*

## 1 INTRODUCTION

Programming languages are one of the most important tools for scientific computing and information processing. At the present time learning programming languages is an obligatory course for the students not only from the discipline of computer sciences but also from other engineering departments.

A programming language contains a number of syntax rules, semantics, design patterns and function libraries. Learners must practice and use them in order to master programming languages. In recent years, a large number of new successful programming languages have been

developed for different needs and applications. In order to help learners of programming languages, a number of web platforms came up supporting learning new programming languages, corresponding technologies and frameworks.

Websites like Stackoverflow<sup>1</sup> offer a question-answer-portal for programmers. The w3schools<sup>2</sup> website aims at teaching web technologies and provides manuals and program examples for different web programming languages, technologies and frameworks. Other web

<sup>1</sup> <https://stackoverflow.com/>, visited on 18.4.2019

<sup>2</sup> <https://www.w3schools.com/>, visited on 18.4.2019

platforms like Kattis<sup>3</sup> and Codeforces<sup>4</sup> provide a huge archive of problems, which can be solved by the user using a programming language of his/her choice. Once the user is of the opinion that (s)he found the solution, the user can upload the solution for validating and rating purposes.

Websites like CodinGame<sup>5</sup> use games for learning programming languages, where learners practice the programming language they want to learn by implementing the game bots using the corresponding language. These games are extra designed with increasing complexity in implementing game bots. Our work also uses the game bots for learning and practicing programming languages. However, as will be seen, our approach has big differences from ones used in these websites.

These existing platforms can help individual users to learn alone. However, they lack of features to support a group of people to learn several programming languages, related technologies and frameworks together, which is typical in the education at universities. For example, the University of Lübeck offers a course in which students learn the Semantic Web technology (including its data model, query language and Ontology concept), Cloud Computing frameworks (including Hadoop, Pig<sup>6</sup>, Spark<sup>7</sup> and Flink<sup>8</sup>), multi-platform programming via the Kotlin<sup>9</sup> programming language and traditional web technologies including HTML, CSS, JavaScript and PHP.

In the lecture, the instructor teaches the principles, concepts and programming paradigms and styles of these programming languages and technologies, and explains them using program examples. In order to really master these languages and techniques, however, students must train themselves by actually using and practicing them a lot. Aiming to provide a supplement to the lecture, in this work we develop the concept of multi-games and introduce a multi-game platform, which helps students to learn programming languages and corresponding technologies easily and efficiently. A multi-game is a video game, which can be decomposed into several mini-games. Each mini-game is to be solved by implementing a program for a game bot. Designing and implementing small games are easier than those tasks for a big game reducing the task complexities for the students. With our multi-game platform, each mini-game can be played independently from the other mini-

games. After solving all mini-games, their implemented game bots work together to play the whole multi-game.

In this work, a video multi-game called “Builders” is developed, which consists of three mini-games. In order to play a mini-game, students need to develop a bot, and different bots are implemented using different languages and technologies. Therefore, when playing the multi-game Builders, students can learn and practice different programming languages and technologies. Furthermore, our multi-game platform is a multi-player platform, where bots from different students can play against each other. Therefore, we call the type of our system a *Multi-Game Platform for Code-Duels*. In our platform, a tournament can be run, where the mini-game bots of students compete with each other in order to obtain a ranking of their bots. The feature of tournament promotes the interest in and engagement with learning and practicing programming languages of students.

Besides the scenario of using the Multi-Game Platform for Code-Duels within lectures there are many more applications of the platform. Of course, the platform can be used in similar scenarios as in lectures whenever new programming languages are learned by the users themselves (*self-study*), the knowledge about these programming languages and their programming patterns have to be deepened, whenever programming languages are learned at school, in the Kindergarten or during on-the-job training. These scenarios of the platform include further to test the qualifications of software developers within the process of human resources recruitment, or to propose challenges to determine “The Software Developer of the Year” or similar within a company. If the bots to be developed solve very complex problems, then the platform might be also used to distribute software development tasks within a company or even to world-wide users (*software crowdsourcing*).

Our contributions include:

- We introduce the concept of multi-game. A multi-game consists of several mini-games, which can be implemented independently using different languages. Decomposing a single game into mini-games reduces the development efforts and the implementation complexity of the game.
- We design the multi-game “Builders” for learning multiple languages. It consists of three mini-games and each mini-game bot can be implemented using different languages.
- We develop an e-learning platform called *Multi-Game Platform for Code-Duels* for learning different languages by playing the multi-game Builders.

<sup>3</sup> <https://open.kattis.com/>, visited on 18.4.2019

<sup>4</sup> <http://codeforces.com/>, visited on 18.4.2019

<sup>5</sup> <https://www.codingame.com/start>, visited on 26.4.2019

<sup>6</sup> <https://pig.apache.org/>, visited on 18.4.2019

<sup>7</sup> <https://spark.apache.org/>, visited on 18.4.2019

<sup>8</sup> <https://flink.apache.org/>, visited on 18.4.2019

<sup>9</sup> <https://kotlinlang.org/>, visited on 18.4.2019

- Three different mini-games of Builders are played independently of other mini-games.
  - Three mini-games work together to form the multi-game Builders.
  - Our platform provides the necessary functionality (e.g. design templates, code debug, test cases), which enable the development of game bots easily.
  - Our platform supports a tournament, where game bots developed by the students can compete against each other. The feature increases the learning interests and engagement of students.
- We perform an user evaluation over the multi-game approach and the platform, and a very positive rating is obtained.

## 2 RELATED WORK

S. Deterding et al. define in [6] the term “gamification” as “the use of game design elements in non-game contexts”. By using gamification producers hope to increase user interests and acceptance of their products. In addition, they use various elements and levels of gamification, such as the distribution of rewards as an element in the first level or the use of a playcentric design as one element of the last level. Our proposed system can be also mainly classified into this last level of gamification due to providing a code-duel platform for students in a university course. However, we implement and visualize also elements of the first level, i.e. rewards are given by visualizing the skill level of a student, where the student achieves higher levels of skill after solving the exercises. Direct competitions between the developed game bots are further E-Learning concepts which have been implemented as tournament in the end of the university’s course. Our platform supports a help system by offering a chat to discuss questions of the students between themselves or a supervisor.

Game design techniques are e.g. experience and level systems (e.g., [13]), where with increased user experience new features or previously hidden parts of the game are unlocked, or based on good performances the user reaches higher levels with increased difficulty or higher standing. Time limits may also be used in order to increase concentration for and engagement with a task. Of course, tasks can be packaged in games or formulated similar to game descriptions.

Game design techniques are not only techniques occurring in games, but also game-accompanying concepts. Players are rated based on their performances, and rating results are listed in e.g. leaderboards (e.g., [5]). Games also often distribute rewards like emblems, badges and prizes whenever a player

reaches a certain performance or plays frequently. Furthermore, visualization elements are provided in games, e.g. progress bars [9] which show the progress until completing a goal. Such elements provide users with the information about their stepwise success in long-term tasks. Another component is direct feedback. If users are directly getting a (positive or negative) feedback, there is a positive impact on their learn efficiency and their engagement [17].

Gamification has been applied into almost all areas of life: online shops [15], advertisement or market research [24], fitness as well as educational programs [6].

The authors in [21] state that learning to program is a complicated task and equips students with problem-solving skills, including the decomposition of problems, developing algorithms (using software patterns) and finally coding. [7] introduces and compares some online courses for learning programming languages aiming at helping their users during the learning process. The investigated platforms include Codecademy<sup>10</sup> and Coursera<sup>11</sup>. Codecademy focuses on web technologies and uses a small-step task system. Coursera offers courses from different schools or universities and certificates are issued to the learner once (s)he finishes a course. However, these course are much more time-consuming. Also research prototypes introduce concepts for learning programming languages online [4], mobile [8, 19] or as desktop application [1].

The article [22] explains the concept of code duels and review systems for improving programming skills by using various rewards. For example, CodeAbbey<sup>12</sup> uses a title system and rewards solved tasks with new prizes, while in CodinGame<sup>13</sup> the programmed game bots can reach higher leagues after winning some duels. Codeforces and Kattis use puzzles with different topics like politics, films, games, sports and other current topics of public’s attention. A ranking list is used in all these platforms for learning programming languages to increase the motivation of their participants. In Kattis you can even compare yourself with participants of your own university, and there is a ranking of the best universities. CodinGame distributes additional badges, which symbolize the learning progress of a participant.

The authors in [18] validate gamification websites for learning programming languages in terms of increased study performance (effectiveness and motivation). Even children in Kindergarden learn coding by using special websites for learning (visual) programming languages<sup>14</sup>

<sup>10</sup> <https://www.codecademy.com/>, visited on 18.4.2019

<sup>11</sup> <https://www.coursera.org/>, visited on 18.4.2019

<sup>12</sup> <http://www.codeabbey.com/>, visited on 18.4.2019

<sup>13</sup> <https://www.codingame.com/>, visited on 18.4.2019

<sup>14</sup> The authors used the ScratchJr programming environment available at <http://www.scratchjr.org/> for their study.

[23]. Agile software development techniques like pair programming have also been investigated for websites for learning programming [16].

The effectiveness of gamification in the e-learning area and other environments has been investigated in various studies. Barata et al. study in [2] and [3] the effects of gamification on an university course. [2] describes the implementation of this course and analyzes the results of a single semester with this system. [3] presents a study over two consecutive semesters and compares the results of this test series with data from previous years (where gamification has not been applied to the university course). The authors in [20] identified a set of digital competency skills like cognitive, language, collaborative and creative problem-solving skills for pupils learning programming.

Positive effects of gamification can be measured in many other areas such as fitness, work and data collection. Most often, users' interest in the respective platforms is increased [14]. Effects of gamification on business, electronic commerce and online-surveys are examined in [24] and [15]. Hamari et al. [14] provided a systematic review of popular literature on gamification by grouping the individual sources according to various criteria, discussed the results of the studies and tried to find common ground.

Different from these contributions and platforms described in this section, our e-learning platform helps learners to learn different programming languages by a new *multi-game* concept. In our concept, a multi-game is divided into several mini-games. Each mini-game has to be solved by implementing a game bot. All mini-games are played separately, but after solving all mini-games, their bots work together to play the whole multi-game. Designing several small bots has less development efforts and implementation complexity than designing a single, big game. Furthermore, our platform supports the feature of a tournament, with which game bots from different students can play against each other. This feature promotes the fun and engagement with learning.

### 3 MULTI-GAME FOR CODE-DUELS

In this research work, we develop a platform of multi-player video games called *Multi-Game Platform for Code-Duels* aiming at helping students to learn different programming languages, techniques, frameworks and tools. In order to play the code-duel video game, a student (i.e., the human player) needs to develop a program for the game bot, which will simulate the artificial intelligence to play the game against the bots from other students or against bots provided by the platform itself.

When playing a video game, players have to make decisions in various different situations, contexts and based on different types of input. Programming such artificial intelligence is usually not straightforward, and thus a bot, which simulates the behavior of a human player, will be quite complex. In order to reduce the complexity of bots and make it easier for students to learn, we decompose a game into several mini-games. The bots of the mini-games can be developed and the mini-games played independently of other mini-games. After all game bots have been developed by the students, they are composed together to build the game bot of the whole multi-game playing the overall game. We call such a game *multi-game*, and the concept of multi-game can be loosely defined as follows:

**Definition (Multi-Game):** A multi-game consists of mini-games. Each mini-game is to be solved by implementing a game bot independently from the other mini-games. All the game bots of the mini-games form together the game bot of the multi-game.

Using multi-games in learning programming languages has a number of advantages. It is not necessary to write a big, complex bot for playing the whole game. Students only need to develop a small bot for each mini-game. Thus bots are relatively easy to implement, however practicing small programs are not reducing the effects of learning at all. On the contrary, a small bot can be more easily developed and thus quickly used to play the mini-game. This will no doubt improve the learning interests and practicing engagement of students.

The rest of this section is organized as follows: before presenting our e-learning platform for multi-games for code-duels, we first identify the requirements of an e-learning platform for learning programming languages, related techniques, frameworks and tools in 3.1. We then present the overview of the architecture of our multi-game platform for code duels to learn and practice several programming languages and associated technologies simultaneously in Section 3.2. The Section 3.3 presents our multi-game for code-duels called "Builders", where each mini-game is described in detail.

#### 3.1 Requirements of E-Learning Platform

Our research work aims at developing an e-learning platform, which will help learners to learn and practice multiple programming languages and techniques. Such an e-learning platform should be easy to use and can provide efficient help in learning. Before designing the e-learning platform for Multi-Games for Code-Duels, we first identify the requirements, which such a platform for learning programming languages should meet, as

follows:

1. The e-learning platform must be easy to use:
  - The students can use the e-learning platform without any software installation.
  - The students can learn the numerous programming languages, techniques, frameworks and tools within one application.
  - The students can enjoy state-of-the-art programming comfort like syntax highlighting, auto completion and displaying clear error messages.
2. The e-learning platform should increase students' engagement:
  - Students should get feedback about the quality of their solution (besides a 'solved' status).
  - The solutions of the students should compete against each other, such that there will be a ranking of the solutions. The rankings might be presented to the students in an anonymous way (where students know their own rankings but not the ones of their fellow students).
  - After submission of solutions and after competing with other solutions the students should get feedback, such that they can improve their own solutions for resubmission.
3. There should be some 'kind of story' over roughly the half year the course takes place.
  - Single exercises over the whole course should be assembled in a way such that their solutions together solve a bigger exercise. However, the single exercises should be still solvable independent from each other.
  - Improving the solutions of the single exercises should improve the solution for the overall bigger exercise.
  - A final ranking of the solutions should be based on the bigger exercise.
4. The e-learning platform should be designed in a way that hinders questionable practices of students.
  - The single exercises should be solvable using different programming languages, techniques, frameworks and tools. With this feature, the lecturer can ask the current students to use a language and related techniques different from one used in the same course in the past years. In

this way, the programming code developed by the students from previous courses can not be used in the current course. Each student must practice the knowledge learned in the course on her/his own.

- The e-learning platform should be designed according to state-of-the-art privacy and security principles in order to avoid hacking and copying of solutions, and should offer a safe platform to run the students' code.

As will be seen, our e-learning platform of Multi-Games for Code-Duels meets these requirements.

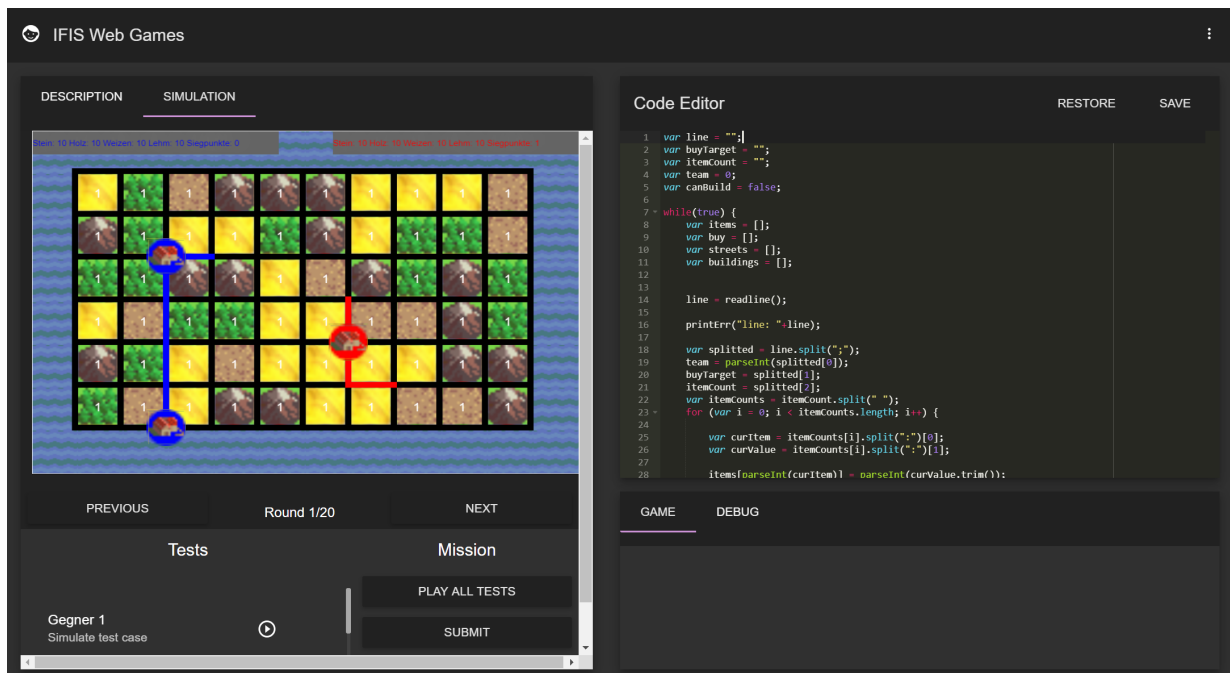
### 3.2 Platform of Multi-Games for Code-Duels

Our multi-game platform for code-duels has been developed as web application for the following reasons:

- Users can use our platform without installing any special software (besides the Internet browser, which is usually pre-installed).
- Users can use our platform anywhere and anytime with any typical client hardware (desktop pcs (recommended), tablets or even smartphones).
- The lecturer always has the possibility to look at the submitted solutions of the users.
- The lecturer can administer submission phases and the tournament through the web frontend.
- The administrator can maintain (e.g. update) the software easily at the backend computers (without the need to update the clients' software).

The frontend of our platform of Multi-Games for Code-Duels is a web user interface (see Figure 1), and consists of following components:

- **A video multi-game "Builders":** Our game is inspired by the board game "The Settlers of Catan" [10]. The map of the game world is presented on the left in Figure 1. It includes three mini-games: Controller, Planner and Executor. In order to play the multi-game, students need to program a bot for each mini-game using required programming and query languages, and related technologies.
- **A code editor:** Students can write and develop the code with the built-in code editor (on the right of Figure 1). It also provides the code templates for implementing the bots. Apart from defining input and output variables, these templates also include the code of basic but tedious operations like parsing the input of a game round (see the Appendix). Using these



**Figure 1:** The web user interface of the platform, where the map of the game world is presented on the left together with buttons to apply test cases and submit. On the right the users can type in the bot's program and look at error and debug messages<sup>21</sup>.

templates, when programming, students can focus on the important functions of the bots.

- **Compiling & Running the Game, Submitting the Code:** In the lower left side of Figure 1, buttons are provided to compile and afterwards run the developed game bot against different test cases. After running a test case, the student can investigate each round of the game and watch the changes in the game map. After all test cases have been successfully run, the student can submit the code to be used as the student's game bot for the tournament.

Figure 2 presents an overview of the architecture of our code-duels platform, which illustrates how the backend of our platform works.

- **Webserver:** controls the user interface and delivers corresponding HTML pages, and handles the authentication process of the platform's users. For these tasks, the webserver accesses the game database where users' data is stored (e.g., users' information and their bot code).
- **Game Service:** When a user performs a test case or submits the bot's code at the frontend of the platform (by clicking the corresponding button), the webserver communicates with the game service to start the game.

The game service will hand over the bot's code to the execution service for compiling and running the bot's code. Afterwards the game service computes the input for the game bot for each game round, delivers this input to the execution service and receives the commands of the game bot from the execution service for this game round. Based on these commands the game service computes the next state of the game, which forms the input for the next game round. This process is repeated until the end of the game. Finally, the game's states of each round is handed over through the webserver to the client's browser for displaying together with the game bot's commands and debug messages.

- **Execution Service:** The bots of the mini-games are compiled and executed by the execution service. For this purpose, depending on the used query or programming language, the SPARQL endpoint is queried or batch files are processed.
- **SPARQL Endpoint:** The SPARQL endpoint is queried whenever the game bot is implemented as SPARQL query. Besides the SPARQL query, the game's current state is transferred from the game service as RDF data, and the query result contains the bot's decisions for the current game round. We use the

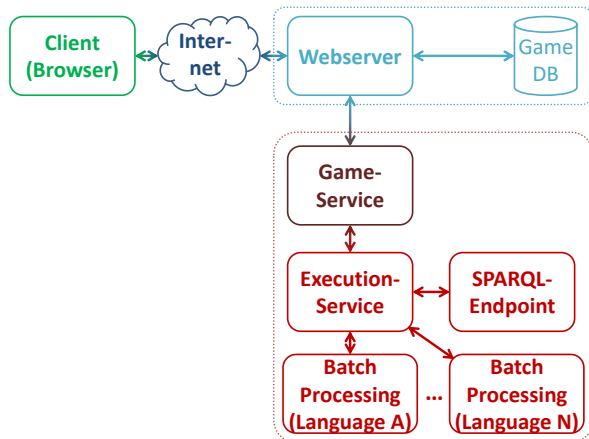


Figure 2: Simplified architecture of the code-duels platform for the multi-game “Builders”

Semantic Web database LUPOSDATE [11, 12] to run the SPARQL endpoint.

- **Batch Processing:** If the game bot is implemented in a general-purpose programming language, then the execution service calls a batch file for this purpose. The batch file contains the instructions for compiling and running the game bot. The input of the game round (as delivered by the Game Service) is handed over to the game bot via standard input and the commands of the game bot are received via the standard output. We currently support JavaScript and PHP as programming languages for the game bots. By communicating via standard input and output the support of other programming languages can be easily integrated (as standard input and output is supported by most programming language environments). If there is any syntax or semantic error detected during compilation, these errors are of course returned back through the game service and webservice to the client for displaying the error text in the client’s user interface.

Standard security approaches have been adopted for user authentication (by providing a login via user and password) and a safe communication in the Internet is guaranteed via SSL certificates and https-protocol. The platform is secured against SQL injection attacks via using prepared statements. In order to hinder malicious bot programs attack the platform to receive important information and sensitive data of the platform (e.g., from the user database), the execution service doesn’t run on the same computer as the webservice and the database management system where the user database is stored. The execution service creates an own folder for each

game bot (implemented in PHP or JavaScript), where the bot’s program is stored and intermediate files for compilation are saved. The users’ rights management of the Linux operating system is used to further enhance the security by running the game bot’s program with the rights of a very restricted user of the server’s Linux operating system. This restricted user only has the rights to read and execute the bot’s program, such that the bot’s program doesn’t have access to other files in the server hosting the execution service. Further security actions may be taken into account like using virtual machines for each bot program or security sandboxes (e.g., as part of SELinux<sup>15</sup>).

### 3.2.1 Performance of our Code Duels Platform

Our operational system is set up as virtual machine with 5 cores and 6 Gigabytes main memory running the Oracle Linux operating system on an Intel Xeon E5620 @ 2.4GHz.

We have measured the execution times of the bot programs in the different languages (i.e., SPARQL, JavaScript and PHP) running on the server. We have used slightly modified versions of the templates in the Appendix as bot programs. The modifications include retrieving the current time at bot program start<sup>16,17</sup> and at the end of the game loop for printing it to the standard error output<sup>18,19</sup>. For a proper running of the template, the JavaScript template was additionally extended by a command for each round<sup>20</sup>. The SPARQL evaluation times were measured by the SPARQL endpoint.

The execution times for running the modified templates (applied to a normal game map of size  $6 \times 10$ ) were measured 10 times. The average execution time for running the modified JavaScript template is 8,316 seconds, the one for the modified PHP template 6,123 seconds and for the SPARQL queries 0,03 seconds. The input data size of the SPARQL queries is 1070 triples and because of significant differences the execution times were measured after a warm-up phase of 10 queries.

The execution times are sufficient for running the test cases and developing the bot programs. The program bots as well as the implemented game are sequential programs. Further performance improvements could be achieved by paralelizing or even distributing program bots and the implemented game. However, as our implemented game is round-based, the algorithmic nature of the game as well as the bot programs is

<sup>15</sup> <https://fedoraproject.org/wiki/SELinux>, visited on 17.4.2019

<sup>16</sup> `var t0=new Date().getTime();` in JavaScript

<sup>17</sup> `$ time_start =microtime(true);` in PHP

<sup>18</sup> `printErr ((t1-t0)+"_milliseconds.");` in JavaScript

<sup>19</sup> `fwrite (STDERR,"{$time_end-$time_start} seconds\n");` in PHP

<sup>20</sup> `print ("BUY_0");`

sequential and is only parallelizable on a fine-granular basis.

The computations of different users running different game bots are completely independent of each other. Hence, the game and the bot programs can be parallelized and even distributed within a cluster. In this way the execution times for the modified templates are minimum time durations for a turn-around time of the students when using our platform of Multi-Games for Code-Duels.

### 3.3 The Multi-Game “Builders”






In this work we develop the multi-game *Builders*. Builders is inspired from the board game “The Settlers of Catan”[10] (released in 1995), but has significant modifications like simpler game rules and a simpler layout of the game world in order to simplify the programming of the game bots. Builders consists of three mini-games: *Controller*, *Planner* and *Executor*. For each mini-game, learners need to develop a bot using a different language. We first describe the overall multi-game in Section 3.3.1 and the three mini-games in the succeeding sub-sections.

#### 3.3.1 Overall Game

The game takes place in a (two-dimensional) game world, which represents a land and is divided into square fields of different types of natural resources. These natural resources are wood, stone, clay and wheat<sup>21</sup>, as depicted in Table 1. The natural resources will be used by players to build other resources like streets and cities, which will in turn bring their Builders natural resources. Each resource field has a value of 1, 2 or 3. When a player builds a city, with each resource field adjacent to the city, the player will acquire the number of resources equal to the value of the field. Players use the natural resources to buy victory points, and the player who acquires the most victory points wins the game.

When playing the game, players can take different actions. They can trade with the bank, build streets and cities, and buy victory points. Victory points are bought using natural resources. Hence players need to own as many natural resources as possible. Natural resources can be acquired by building cities. When a player builds a city, the player receives the resources indicated in the resource fields around the city. In order to build cities, players need to build streets, because a city must be built along a street owned by the player (except for the first city). Building a city costs one piece of wood, clay

**Table 1: Ids and images<sup>21</sup> of the resource fields in the Builders game**

ID	Resource	Image
0	Sea	
1	Stone	
2	Wood	
3	Wheat	
4	Clay	

and wheat, and building a street costs two stones. If a player wants to build a city but (s)he lacks for a piece of a resource like wood, the player can trade with the bank for it. To trade resources, the player offers four resources of one type to obtain a resource of another type, e.g. four stones for a piece of wood. The prices of victory points are randomly decided by the game platform and increase slightly as more game rounds are passed. For example, in the first-round playing, a victory point may cost a stone.

In order to be able to play the game, players need some initial resources. At the very beginning of the game, two resources of each type are allocated to each player. Furthermore, at the beginning of each game round, a resource type is randomly picked by the game platform. If a player has built a city in previous rounds, which is adjacent to such a resource field, the player will receive the resources equal to the value of the field. Figure 3 shows that two players are playing the game. The blue streets and cities are built by one player, and the red ones are built by the other. Let us assume that the resource wheat is selected at the beginning of a game-round. The blue player has built a city, which is adjacent to two wheat fields with the value 1. Hence the blue player receives two pieces of wheat. The red player has a city adjacent to a wheat field with the value 1 acquiring hence a piece of wheat.

Two types of coordinate systems (see the numbers in white color in Figure 3) are used in the game to unambiguously address streets and cities (*building coordinate system*) and resource fields (*field coordinate system*) with integer values. The building coordinate system used for streets and cities is in the upper left corner. For example, the short blue street starts at coordinates (2, 2) and ends at (3, 2) in this system. The field coordinate system for the resource fields is shown in the lower right corner. For example, the wheat field in the lower-right corner of the map locates at the coordinates (10, 6). The periphery in the map of the game world is blue sea fields, which are also addressed by the

<sup>21</sup> Images provided by Andre Mari Coppola under CC BY 4.0 <http://creativecommons.org/licenses/by/4.0/> (visited on 18.4.2019)



field coordinate system. The information of players is displayed in the upper left and upper right corners of the game map, including the quantities of resources and victory points of each player in the corresponding colors.

The overall Builders game contains three mini-games.

**builders.controller:** This mini-game decides whether to buy victory points, and build and trade resources.

**builders.planner:** This mini-game identifies all valid fields for building streets and cities.

**builders.executor:** This mini-game decides where to build streets and cities.

**builders:** the overall game, which consists of the above three mini-games.

Builders can be played as an overall game, or each mini-game can be played independently of other mini-games and the overall game. For this purpose, students need to develop a bot for each mini-game. Table 2 describes the tasks to be implemented by the bots. Once a bot is developed, the student can play the corresponding mini-game. If a student implements all three bots, (s)he can play Builders as overall game with the three bots. How the three bots work together to play the overall game is depicted in Figure 4. The functionality to enable the interplay of the game bots of the three mini-games has been provided by the game, and does not need to be implemented by students.

### 3.3.2 Controller

**Task:** The controller is the decision maker of the game. It decides if the player should trade resources in order to build streets and cities, or build streets and cities in order to obtain more resources or use the natural resources to buy victory points. Such decisions have an important impact on the win chance of the game. In order to make a good decision, the bot must consider various different situations and contexts, e.g., what natural resources are owned, which and where the streets and cities have been built, how many victory points have been acquired and how many rounds have passed on. Therefore, developing a good strategy for decision-making is a challenge for the student while implementing the bot.

**Language:** In the current implementation of our multi-game platform for code-duels, we choose JavaScript as the programming language for the controller bot. However, all general-purpose

programming and script languages<sup>22</sup> are suitable for implementing the bot. In general JavaScript can be used in game development whenever browser-games are developed, but also server-side realizations are possible using the Node.js<sup>23</sup> framework using one language in the browser and in the server (and with the use of frameworks like Cordova<sup>24</sup> and Ionic<sup>25</sup> also for mobile platforms).<sup>26</sup>

**Input:** In order to make decisions, the controller needs to know the current resources of the player, the cities and streets of all players, and the price of victory points. The information is given to the controller by the game service as round input.

In order to simplify exchange of information between the bots and the code-duels platform, standard input and standard output are used. Most programming and script languages support standard input and output natively in a very simple way, such that (i) students can easily program the routines for the communication with the platform, and (ii) new languages can be integrated without changing the routines for communication with the bots.

We hence propose a very simple input format where a line of the input contains all the necessary values, separated by semicolon ‘;’. First, the player’s team is encoded in the line, then the price of the current victory point and the number of resources the player owns. The resource quantities of both fields are encoded in the same way. The resources are separated by a space, followed by the ID of the resource and the quantity, separated by a colon ‘:’. Then the number of cities and coordinates and ownerships of the cities are given, and as last item the number of streets and the streets themselves (i.e., coordinates, direction and ownership).<sup>27</sup>

<sup>22</sup> As a matter of fact, our code-duels platform is designed for an easy exchange of the bots’ implementation languages: It is sufficient to just change the configuration of the code-duels platform accordingly and provide a template with which the students can start. Completely new programming and script languages are easy to integrate due to the modularized backend of the platform.

<sup>23</sup> <https://nodejs.org>

<sup>24</sup> <https://cordova.apache.org/>

<sup>25</sup> <https://ionicframework.com/>

<sup>26</sup> Alternatives for one programming language for all platforms like Kotlin (<https://kotlinlang.org/>) with multi-platform support may be also interesting.

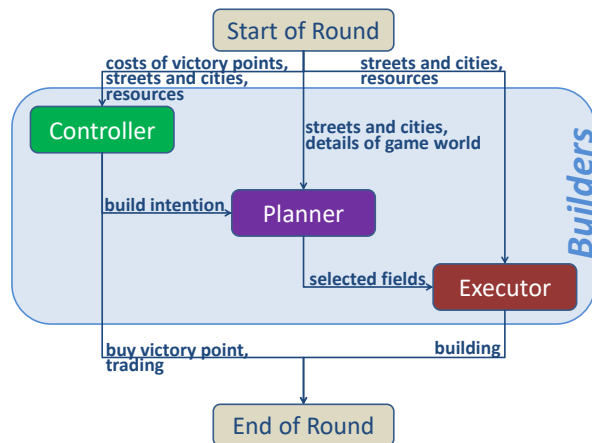
<sup>27</sup> To read the input of the controller and package it into meaningful data structures, many `split` operations on the input line are necessary. Since the learning success was rated as low for the students, and such a task has an increased frustration potential, it was decided to already provide code for the purpose of reading and parsing the round input in the template. The resulting template can be seen in Listing 4 in the attachment “Templates”.



Figure 3: The field coordinate system (in the lower right corner) and the building coordinate system (in the upper left corner) of the game<sup>21</sup>

Table 2: Tasks of the different mini-games to be implemented by the students' bots

Controller	Planner	Executor
<p>The student needs to implement the controller bot, which decides whether or not to buy victory points, and build and trade resources. When developing the strategy for decision-making, the student should consider:</p> <ul style="list-style-type: none"> <li>• When should be streets and cities built?</li> <li>• Which resource is the most desired one to be mined? Close to which resource should a city be built?</li> <li>• Is it always wise to buy victory points? When should victory points be bought?</li> <li>• When and which resource should be traded?</li> </ul>	<p>The planner receives as input a desired resource and should deliver x- and y-coordinates for building a city adjacent to a field with the desired resource. The student's bot does not need to select all valid fields, such that the planner can already filter adverse fields. The student's bot may sort the fields according to some ranking criteria. Invalid fields are e.g. those which</p> <ul style="list-style-type: none"> <li>• are not close to a field with the desired resource</li> <li>• are connected to streets of the opponent</li> <li>• already contain a city</li> </ul>	<ul style="list-style-type: none"> <li>• Selection of a field specified by the planner for building a city</li> <li>• Find a path (from an own street) to this field (not crossing the opponent's streets and cities)</li> <li>• Find existing streets on this path</li> <li>• Build streets, if not already there on the path</li> <li>• Build the city (at the selected field), if the street is connected to an own street and no street was already built in this round</li> </ul>



**Figure 4:** An overview of the communication between each component (of the single mini-games) of the Builders' multi-game

**Listing 1:** Example of controller-input

```
"1;0:1;0:4 1:5 2:3 3:3;3;4:0:0 9:1:1
↪ 9:3:1;3;4:0:LEFTRIGHT:0 9:1:
↪ TOPDOWN:1 9:2:TOPDOWN:1"
```

For example, looking at Listing 1, the following round input is encoded:

- The player has the team number one.
- The current victory point costs a stone.
- The player has four stones, five units of wood, three wheat and three clay (units).
- There are 3 cities on the map of the game world.
- There is a city of team 0 at coordinates (4, 0).
- There is a city of team 1 at coordinates (9, 1).
- There is a city of team 1 at coordinates (9, 3).
- There are 3 streets.
- There is a horizontal street of team 0 to the right starting at coordinates (4, 0) and ending at coordinates (5, 0).
- There is a vertical street of team 1 downward starting at coordinates (9, 1) and ending at coordinates (9, 2).
- There is a vertical street of team 1 downward starting at coordinates (9, 2) and ending at coordinates (9, 3).

**Output:** When the controller makes a decision, it sends the decision as command either to the Game Service or to the Planner bot. The outputs of the controller and their relation are given in Table 3. In a game round, the controller can make a decision to trade resources when a resource is needed for developing new resources. The controller informs the Game Service of the trade decision. The Game Service allocates the wanted resource and meanwhile takes the corresponding resources away from the controller. If the controller does not have enough resources for the trade, the Game Service will not perform the trade command.

In each round, the controller must decide whether to buy a victory point or not. If it decides to buy victory points, it communicates with the Game Service. The Game Service performs the command: either selling the victory points to the player, or not selling victory points if the player does not have enough resources. After the buy command is executed, the current game round is finished. If the controller decides not to buy victory points, it can decide to build resources.

The controller does not make a decision of building streets/cities, instead, it checks and decides which resource needs to be mined by building cities. The planner then selects appropriate fields, the executor builds the streets and cities and the game round ends. If the controller decides to do nothing, it sends the END ROUND command to the Game Service, which ends the current game round and starts a new round.

**Test Case:** We provide a test case for the controller to check whether or not the implemented strategy of the student's game bot is suitable to defeat a weak opponent. The weak opponent in this test case owns only one city, but your team owns already two cities, such that your team receives more resources (with a high probability). In order to pass the test case your team should have bought more victory points than the weak opponent after 20 rounds.

### 3.3.3 Planner

**Task:** When the planner receives the BUILD (resource id) command from the controller, it selects valid fields for building streets and cities that will bring the required resources, and filters out invalid fields, which are not adjacent to a field with the wanted resource, or which are connected to streets of the opponent.

**Language:** As the task of the planner is simple enough to be realized as query, the language of the student's bot is SPARQL, but also other query languages can be easily integrated into the code-duels platform.

**Table 3: The outputs of the controller**

Command	Parameter	Description	Optional	Excludes
BUY	boolean	Should a victory point be bought?	No	
BUILD	Resource id id	Build resource with id id	Yes	BUY true
TRADE	Resource id from, int number of, resource to	Exchange number of resources to with resource from	Yes	
END ROUND		End the current game round	Yes	

**Table 4: Description and syntax of the input data of the planner**

Input Type	RDF Data	Description
Resource Field	<pre>_:a2b1 :x_coord "2"^^xsd:integer;        :y_coord "1"^^xsd:integer;        :resource :stone;        :value "2"^^xsd:integer;        :type :title.</pre>	A resource field at coordinates (2, 1) with resource stone and value 2.
Cities	<pre>[] :x_coord "3"^^xsd:integer;    :y_coord "1"^^xsd:integer;    :nw _:a3b1;    :ne _:a4b1;    :se _:a4b2;    :sw _:a3b2;    :team "1"^^xsd:integer;    :type :building.</pre>	A city of team 1 at coordinates (3, 1) with given adjacent resource fields.
Street	<pre>[] :x_coord "3"^^xsd:integer;    :y_coord "1"^^xsd:integer;    :dir :lr;    :team "1"^^xsd:integer;    :type :streat.</pre>	A street of team 1 starting from coordinates (3, 1) and ending at coordinates (4, 1).
Field for Building	<pre>[] :x_coord "3"^^xsd:integer;    :y_coord "1"^^xsd:integer;    :nw _:a3b1;    :ne _:a4b1;    :se _:a4b2;    :sw _:a3b2;    :type :building_place.</pre>	A building field at coordinates (3,1) with the given adjacent resource fields.
Goal of Building	:buildtarget :resource :wood.	The goal of building in this round is wood.
Own Team	:ownplayer :team "1"^^xsd:integer.	The team of the player is 1.

Students should design a query, which selects all valid fields to build streets and cities by returning their x- and y-coordinates. If no valid fields exist, the result of the query is empty. There are often simple tasks outside of game bots in games, which just need to retrieve some information about e.g. users, available servers and game status. It might be promising to store this information in a database such that queries can be used for these tasks (and to benefit from database features like atomicity and consistency).

**Input:** In order to distinguish valid from invalid fields, the planner needs the input information about all fields, and the coordinates of cities and streets built. The complete input information and its format are listed in Table 4. Named blank nodes are used to describe the resource fields. This has the advantage that cities and streets can be described with these fields. For streets and cities, however, unnamed blank nodes are used because they no longer need to be referenced.

**Output:** After executing the query, the planner outputs a group of x- and y-coordinates of valid fields to the executor, which can be used to build streets and cities.

**Template:** In this case, the skeleton of a SPARQL query is sufficient as template. In order to save the students' typing, some basic information has been provided in the template, e.g. the used prefixes, the selection of the own team and the desired resource. Furthermore, in order to focus learners on the functionality of the bot, triple patterns are provided, which select all the fields for building, their adjacent fields and resources. This template is presented in Listing 3 in the Appendix "Templates".

**Test Case:** Constructing suitable test cases for the planner is relatively simple. It has to be tested that the students' queries return correct fields for different resource requirements. In addition, cities and streets should be inserted at different coordinates to test that these coordinates should be ignored. Figure 5 presents such a situation. In addition to own streets and cities (in blue) there is an enemy city (in green) with some streets. The result of the template given in Listing 3 is presented in Figure 6. As the template does not make any restrictions (and returns every field), every possible field is marked with an arrow. Valid issued fields are marked by a green arrow, invalid ones by a red arrow.

### 3.3.4 Executor

**Task:** The executor bot is responsible for finding the suitable fields to build streets and cities in order to develop the resources required by the controller. In this mini-game students develop and implement a simple path-finding algorithm. Streets can only be built between fields where at least one of their own streets or cities is located. In addition, newly built streets may not lead to the fields connected by an opposing street. Newly built cities must always be located at coordinates that are reached with its own street. The exception for this rule is the first city, which may be built at freely chosen coordinates.

**Language:** The implementation of path-finding algorithms need the features of general purpose programming languages like loops, recursion and so on. For the executor we decided to use the PHP programming language, but any other general-purpose programming language may be used (after integration of its support into the platform). PHP is most widely used as server-side programming language and hence for the game development in general may be used for the server-side code (not only restricted to game bots).

**Input:** The executor receives already built cities and streets, the current resources of the player and the fields selected by the planner as inputs. Similar to the input of the controller, the executors' input consist only of one line. Different values are separated by semicolon ';'. First, the available resources are given, each separated by a space, followed by the id of the resource and its quantity, separated by a colon ':'. The number of cities and the values of the individual cities are then handed over. These are represented by three values, the x- and y-coordinates of the city and the owner's team. Afterwards the line contains the number of streets. Each street is represented by four values, the x- and y-values of starting coordinates, the direction of the street, and the owner's team.

Listing 2 provides an example of this encoding, containing the following information:

- The player owns one stone, two wood, four wheat and three clay units.
- There is one city on the map.
- There is a city of team 1 at coordinates (9, 3).
- There are two streets on the map.
- Team 1 owns a street downward starting at coordinates (9, 2).



Figure 5: Example test case for the planner<sup>21</sup>

- Team 1 owns a street downward starting at coordinates (9, 1).
- The planner provides a coordinate for building.
- The planner suggests to build at the coordinates (9, 1).
- The player's team has the team number 1.
- The map's width is 10.
- The map's length is 6.

**Output:** The output of the executor is listed in Table 5. After finding suitable fields for building streets and cities, it outputs a command and informs the Game Service to build them at the corresponding coordinates. The Game service performs the building task or declines it if the given fields are invalid or the player doesn't have enough resources. It then ends the current game round and starts a new round if the game isn't finished yet.

**Test Case:** For testing the students' bots, at least three types of test cases should be created. One should test that the program works when passing only a few fields. Another should check that a number of fields do not overwhelm the program. The last type of cases should

#### Listing 2: Example input of the executor mini-game

```
"0:1;1:2;2:4;3:3;1;9:3:1;2;9:2:
↔ TOPDOWN:1;9:1:TOPDOWN:1;1;9:1;1;10
↔ ;6"
```

test that enemy streets and cities are being bypassed. Especially for the last type, different test cases can be generated to intercept possible special cases.

**Template:** The template is presented in Listing 5 in the Appendix "Templates". The tedious and error-prone `split` operations are given in the template.

## 4 EVALUATION

This section presents the user evaluation of our code-duels platform for the multi-game "Builders". The evaluation contains two parts: an online questionnaire and playing the game by developing the bots. The evaluation is performed in the University of Lübeck during the period from 19.12.2017 until 01.01.2018. The participants are recruited freely by sending an e-mail to



Figure 6: The result of the template of Listing 3 (test case “wheat”)

Table 5: Description and Syntax of the output of the executor

Command	Parameter	Description	Optional	Excludes
BUILD street	int x, int y, string direction	Build street at coordinates (x,y) in the given direction	Yes	BUILD city
BUILD city	int x, int y	Build city at coordinates (x, y)	Yes	BUILD street

the MINT section of the University of Lübeck.

At the end of the evaluation period, eight participants completed the questionnaire. 31 students participated in the development of bots and played the corresponding mini-games or the overall game. Among them, five students successfully developed the controller mini-game, four students programmed the planner, and two students solved the executor. Finally two students implemented all three mini-games, and one of them defeated the boss in the overall multi-game.

#### 4.1 Questionnaire

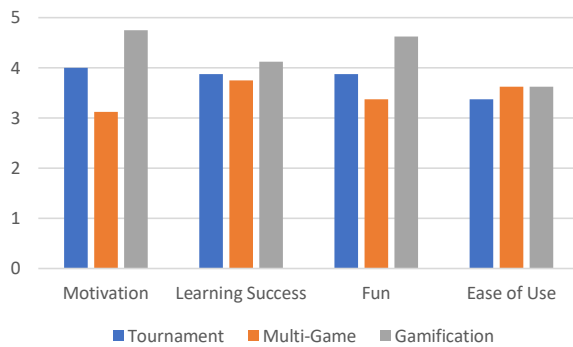
The questionnaire contains three parts: (i) General demographic questions about existing programming

knowledge; (ii) Rating how the features and design of the code-duels platform impact the learning of programming languages; (iii) Providing feedback about the features and design of the platform. The questions of the first two parts are rated with a scale from 1 (negative) to 6 (positive), and in the third part participants provide feedback in free text.

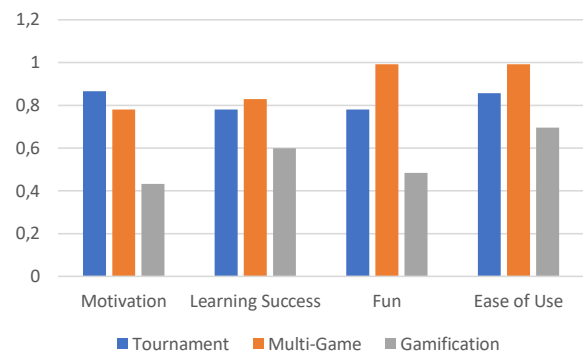
##### 4.1.1 Demography

The first part of the questionnaire contains general demographic questions about existing programming knowledge of participants.

First, the users were asked about their general programming skills. Only one user stated that (s)he



**Figure 7: Mean of the participants' answers**



**Figure 8: Standard deviations of the participants' answers**

had no prior knowledge. Five of the users rated their programming skills with 4 or 5 to be good. The remaining two judged their knowledge with a 3.

Subsequently, they were asked about knowledge of web technologies, e.g., query and web programming languages. Four out of eight participants had no any knowledge about the web technologies. Three users rated their knowledge as good with a 4 or 5. Only one user rated this question with a 3. Three of the participants had experience with e-learning platforms, two of which use them regularly with a rating of 4 or 5. The third uses these systems rarely with a rating of 2.

Finally, the users were asked about their knowledge of the programming languages, i.e., PHP, JavaScript and SPARQL, which are implementation languages of the mini-games of Builders. Four users did not have any knowledge about JavaScript language and the other four did not know about PHP. Only one of them had knowledge about SPARQL and this user rated her/his SPARQL knowledge with 5 to be good. Two users rated their JavaScript skills to be good, three rated their PHP skills to be good.

#### 4.1.2 Gamification, Multi-Game and Tournament

In the second part of the questionnaire, participants assess the impact of the game features (gamification, multi-games and tournament) provided in our code-duels platform to learning programming languages, including the motivation of learning, the self-esteemed learning success, the fun of using the system to learn and the ease of use of the system for learning. Figure 7 presents the mean value of the participants' answers, and Figure 8 provides the standard deviations of the participants' answers.

The evaluation results show that the game features (gamification, multi-games and tournament) have a quite positive impact on the learning of programming

languages, and they have a similar degree of impact on the learning, but the multi-game concept obtains a slightly lower rating. In the evaluation, the students only need to develop three small bots. If the students are also asked to develop a single bot for the overall Builders game, and compare the implementation complexity and development efforts, we are sure that the feature of the multi-game divided into mini-games would be rated much better, such that the multi-game is an important method in learning. In addition, it should be noted that the evaluations of the multi-game concept have relatively high standard deviations compared to those of the other aspects. It therefore seems to be highly dependent on each person's opinion whether this concept is perceived as positive or negative.

#### 4.1.3 Platform Design

At the end of the survey, students are solicited to assess the features and design of our platform and give feedback in free text. The assessment was generally rather positive. Students also gave valuable comments and suggestions for improvement, which are presented in this part.

The biggest criticism were the descriptions of the mini-games and the connection with the multi-game. Several times it was noted that it was not clear enough what exactly the tasks of each mini-game was. In addition, two errors and minor inaccuracies were reported in the description of the planner mini-game. The descriptions have already been significantly improved (by e.g. adding more information, to-do-lists for students for each mini-game and figures providing an overview of the connections between the different mini-games). The errors have been corrected based on this feedback.

As a second negative point, unclear error messages



were noted. Many users were not always sure if there was an error in their source code or not. This was compounded by the fact that an error occurred during the test that stopped the execution of the solution without an error message. We already worked over the platform's code improving error handling.

Many commented that programming skills in SPARQL would be needed to solve the planner mini-game. Our platform and the multi-game Builders will be used within a university course containing Semantic Web topics including the SPARQL query language. Hence the students will have knowledge about SPARQL to solve the planner mini-game.

As the quality of each mini-game's bot also influences the quality of the overall multi-game bot, one user criticized that it is demotivating if (s)he solves one mini-game only badly. This would reduce the motivation to improve other mini-games, too.

Several students assess that this game provides a good exercise for learning programming languages, and decomposing a game into several mini-games reduces overall complexity without limiting the user's capabilities. When asked whether the e-learning platform is a supplement to the university course, five out of eight responded with a clear 'yes'. Two answered 'more yes' and only one user with 'maybe'.

On average, 6 students spent each 1.2 hours to implement the controller. 4 students finish the planner and they spent an average of 2.5 hours each with this mini-game. Only 2 users implemented the executor mini-game, one of whom spent 4 hours, the other 1 hour.

The feedback provided by the students on the invested time basically coincides with the expectations. In particular, the implementation of a route finding algorithm can take a long time. Also, the incorporation into the RDF format of the planner can take a lot of time. Only the implementation of the controller took longer time than expected. One reason could be the description, which we have significantly improved based on the feedback. Another reason could be that the controller was the mini-game to start with and hence students may have needed some time to get used to the overall user interface and game.

## 4.2 Analysis

In summary, the general assessment of the implemented concepts was more positive than negative. It can already be stated that the concepts of multi-games and the tournament have a positive effect on the fun and motivation of the students. In particular, the feedback from the user who has already participated in the corresponding university course has been extremely positive.

The participants, who have little to no experience with web-based programming languages, evaluated very positively. This shows that the students who didn't visit a lecture can use the platform to catch up on missing material.

Although it was noted that it is difficult to solve the problems without prior knowledge, mini-games were still solved by students who had little or no experience with the programming and query languages (especially with SPARQL). We hence conclude that relevant units of the lecture can be learned with the help of the platform with comparatively little time effort.

When implementing the bots, students use different approaches to the tasks. Of the submitted solutions for the controller mini-game, four used an approach that first built streets and cities for a certain number of rounds and victory points were bought subsequently. A solution bought victory points whenever possible and tried not to build, but this user did not pass all mini-games to compete in the overall multi-game. Victory points were often bought after a certain condition was fulfilled. These conditions were 3 built cities, 4 built cities, 3 rounds or 8 rounds. Only one implementation used the trade option to buy victory points if only one resource was missing.

There were four solutions for the planner mini-game, and all used an approach in which all valid fields were returned. Only one approach sorted the fields by x- and y-coordinates. Three solutions used `FILTER EXISTS` or `FILTER NOT EXISTS`, sometimes in combination with `OPTIONAL`. Only one solution applied the `MINUS` operator.

Two students finished the executor mini-game. One used a brute-force approach to path-finding, where runtime overhead was reduced by considering the costs of an already found path as upper limit to early stop the execution for more costly paths. The second student used a hard-coded approach, which was tailored to one test case to be passed for the submission of the bot. To prevent hard-coded approaches in the future, we already provide more test cases, but also hidden or randomized test cases could be built in.

## 5 SUMMARY AND CONCLUSIONS

In this work, we developed an e-learning platform called *Multi-Game for Code-Duels*, aiming to help students to learning different programming languages efficiently and easily. In order to realize this aim, a concept of multi-game is developed and used in the platform. A multi-game consists of several mini-games, where each mini-game is solved by developing a game bot. Each mini-game is independent from the other mini-games, and all the mini-games can work together to form a

big multi-game. Developing several small mini-games, instead of a single big game, reduces the development efforts and implementation complexity of the game bots significantly.

In this platform, we develop a multi-game *Builders*, which is inspired by the board game Settlers of Catan[10]. The Builders multi-game consists of the three mini-games *Controller*, *Planner* and *Executor*. For each mini-game, a bot needs to be implemented using a different language. One small bot of a mini-game can be more easily and quickly developed than a big bot for the whole multi-game. Once a mini-game bot is developed, the student can use it to play the corresponding mini-game no matter whether or not the student has developed the bots of other mini-games. This will in turn bring the student more fun and engagement with learning and practicing programming.

Finally, a user evaluation was carried out. The users provide quite positive ratings to the e-learning platform of Multi-Game for Code-Duels to help the learning of programming languages. Furthermore, many of the users had little experience with web-based programming languages and especially SPARQL queries, but they could develop the corresponding bots using our platform. This shows that our multi-game platform for code-duels can help students to learn programming and accompany the courses of programming languages.

Currently, our platform supports JavaScript as the implementation language for the controller, SPARQL for the planner and PHP for the executor. However, other languages can be easily integrated into our multi-game platform. Future work is the integration of other types of languages and frameworks such as distributed programming models, e.g. MapReduce<sup>28</sup>, script languages for big data processing like Pig<sup>29</sup>, frameworks for modern cloud databases (e.g., Hive<sup>30</sup>), and even languages for describing content and its layout like HTML and CSS. We expect to obtain new research results when integrating and utilizing those languages in our Multi-Game platform for Code-Duels, which are not general-purpose programming languages.

Furthermore, development of other types of multi-games, which are e.g. not round-based and having real-time requirements, and development of commercial multi-player online games (based on the human players' view) are other directions for future work.

<sup>28</sup> <https://wiki.apache.org/hadoop#MapReduce>, visited on 18.4.2019

<sup>29</sup> <https://pig.apache.org/>, visited on 18.4.2019

<sup>30</sup> <https://hive.apache.org/>, visited on 18.4.2019

## REFERENCES

- [1] M. Al-Shawwa, I. Alshawwa, and S. Abu-Naser, "An intelligent tutoring system for learning java," *International Journal of Academic Information Systems Research (IJAIRS)*, vol. 3, no. 1, pp. 1–6, 2019.
- [2] G. Barata, S. Gama, J. Jorge, and D. Gonçalves, "Engaging engineering students with gamification," in *Games and virtual worlds for serious applications (VS-GAMES), 2013 5th international conference on*. IEEE, 2013, pp. 1–8.
- [3] G. Barata, S. Gama, J. Jorge, and D. Gonçalves, "Improving participation and learning with gamification," in *Proceedings of the First International Conference on gameful design, research, and applications*. ACM, 2013, pp. 10–17.
- [4] S. Bhatti, A. Dewani, S. Maqbool, and M. A. Memon, "A web based approach for teaching and learning programming concepts at middle school level," *International Journal of Modern Education and Computer Science (IJMECS)*, vol. 11, no. 4, pp. 46–53, 2019.
- [5] C. Cheong, F. Cheong, and J. Filippou, "Quick quiz: A gamified approach for enhancing learning," in *17th Pacific Asia Conference on Information Systems (PACIS)*, 2013.
- [6] S. Deterding, D. Dixon, R. Khaled, and L. Nacke, "From game design elements to gamefulness: defining gamification," in *Proceedings of the 15th international academic MindTrek conference: Envisioning future media environments*. ACM, 2011, pp. 9–15.
- [7] L. M. Dubowy, "Coden für alle – programmieren lernen mit online-kursen," *c't*, vol. 18, pp. 124 – 129, Aug. 2015.
- [8] A. S. Filipe and A. Nobre, "Design of a learning framework for open mobile applications," *Educação em Foco*, vol. 24, no. 1, pp. 513–530, 2019.
- [9] D. R. Flatla, C. Gutwin, L. E. Nacke, S. Bateman, and R. L. Mandryk, "Calibration games: Making calibration tasks enjoyable by adding motivating game elements," in *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, ser. UIST '11, 2011, pp. 403–412.
- [10] C. GMBH, "Catan - Das Spiel — Catan.de," <https://www.catan.de/spiel/die-siedler-von-catan>, visited on 26.4.2019.
- [11] S. Groppe, *Data Management and Query Processing in Semantic Web Databases*. Springer Verlag, Heidelberg, 2011.

- [12] S. Groppe, “Luposdate semantic web database management system,” <https://github.com/luposdate/luposdate>, visited on 14.4.2019.
- [13] J. Hamari and J. Koivisto, “Social motivations to use gamification: An empirical study of gamifying exercise,” in *21st European Conference on Information Systems (ECIS)*, 2013.
- [14] J. Hamari, J. Koivisto, and H. Sarsa, “Does gamification work?—a literature review of empirical studies on gamification,” in *System Sciences (HICSS), 2014 47th Hawaii International Conference on*. IEEE, 2014, pp. 3025–3034.
- [15] G. Heinemann, *Der neue Online-Handel: Geschäftsmodell und Kanalexzellenz im Digital Commerce*. Springer-Verlag, 2017.
- [16] O. Iskrenovic-Momcilovic, “Pair programming with scratch,” *Education and Information Technologies*, vol. 24, no. 5, p. 2943–2952, 2019.
- [17] K. M. Kapp, “Games, gamification, and the quest for learner engagement,” *T+ D*, vol. 66, no. 6, pp. 64–68, 2012.
- [18] F. L. Khaleel, N. S. Ashaari, and T. S. M. T. Wook, “An empirical study on gamification for learning programming language website,” *Jurnal Teknologi*, vol. 81, no. 2, 2019.
- [19] S. I. Malik, R. Mathew, R. Al-Nuaimi, A. Al-Sideiri, and J. Coldwell-Neilson, “Learning problem solving skills: Comparison of e-learning and m-learning in an introductory programming course,” *Education and Information Technologies*, vol. 24, no. 5, p. 2779–2796, 2019.
- [20] J. Nouri, L. Zhang, L. Mannila, and E. Norén, “Development of computational thinking, digital competence and 21st century skills when learning programming in k-9,” *Education Inquiry*, pp. 1–17, 2019.
- [21] P. Piwek, M. Wermelinger, R. Laney, and R. Walker, “Learning to program: from problems to code,” in *Proceedings of the 3rd Conference on Computing Education Practice*, 2019.
- [22] H. Schulz, “Wetthacken – spielerisch programmieren üben im netz,” *c’t*, vol. 18, pp. 130 – 133, Aug. 2015.
- [23] A. Strawhacker and M. U. Bers, “What they learn when they learn coding: investigating cognitive domains and computer programming knowledge in young children,” *Educational Technology Research and Development*, vol. 67, no. 3, pp. 541–575, 2019.
- [24] E. Uzunova *et al.*, “A new business model of marketing research,” *Izvestiya*, no. 1, pp. 32–42, 2015.

## AUTHOR BIOGRAPHIES



**Sven Groppe** earned his diploma degree in Informatik (Computer Science) in 2002 and his Doctor degree in 2005 from the University of Paderborn. He earned his habilitation degree in 2011 from the University of Lübeck. He worked in the European projects B2B-ECOM, MEMPHIS, ASG and TripCom. He was a member of the DAWG

W3C Working Group, which developed SPARQL. He was the project leader of the DFG project LUPOSDATE, an open-source Semantic Web database, and one of the project leaders of two research projects, which research on FPGA acceleration of relational and Semantic Web databases. He is currently one of the project leaders of two DFG projects focusing on GPU acceleration of database indices and Semantic Internet of Things. He is also the chair of the Semantic Big Data workshop series, which is affiliated with the ACM SIGMOD conference (so far 2016 to 2019), and of the Very Large Internet of Things workshop in conjunction with the VLDB conference (so far 2017 to 2019). His research interests include databases, Semantic Web, query and rule processing and optimization, Cloud Computing, acceleration via GPUs and FPGAs, peer-to-peer (P2P) networks, Internet of Things, data visualization and visual query languages.



**Ian Pösse** was born in Aschaffenburg, Germany, in 1995. He earned his bachelor's degree in Computer Science in 2018 at the University of Lübeck. His bachelor thesis covered the development of a gamified e-learning platform for webbased programming languages. He is currently continuing his study aiming for

a master's degree.

## APPENDIX: TEMPLATES

This Appendix contains the templates for the students' bots.

**Listing 3: SPARQL query template for the planner**

```
1 PREFIX : <http://ifis.uni-luebeck.de/builders/>
2 PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
3
4 SELECT DISTINCT $x $y
5 WHERE
6 {
7     :buildtarget :resource $resource .
8     :ownplayer :team $ownteam .
9
10    $place :ne $field_ne .
11    $field_ne :value $v_ne .
12    $field_ne :resource $r_ne .
13
14    $place :nw $field_nw .
15    $field_nw :value $v_nw .
16    $field_nw :resource $r_nw .
17
18    $place :se $field_se .
19    $field_se :value $v_se .
20    $field_se :resource $r_se .
21
22    $place :sw $field_sw .
23    $field_sw :value $v_sw .
24    $field_sw :resource $r_sw .
25
26    $place :x_coord $x .
27    $place :y_coord $y .
28
29    # TODO: Add game logic here
30
31 }
```

Listing 4: JavaScript code template for the controller

```

1 var line = "";
2 var buyTarget = "";
3 var itemCount = "";
4 var team = 0;
5 var round = 0;
6 while(true) {
7     var items = [];
8     var buy = [];
9     var streets = [];
10    var buildings = [];
11    line = readline();
12    var splitted = line.split(";");
13    team = parseInt(splitted[0]);
14    buyTarget = splitted[1];
15    itemCount = splitted[2];
16    var itemCounts = itemCount.split("_");
17    for (var i = 0; i < itemCounts.length; i++) {
18        var curItem = itemCounts[i].split(":")[0];
19        var curValue = itemCounts[i].split(":")[1];
20        items[parseInt(curItem)] = parseInt(curValue.trim());
21    }
22    var buyTargetSplit = buyTarget.split("_");
23    for (var i = 0; i < buyTargetSplit.length; i++) {
24        var curItem = buyTargetSplit[i].split(":")[0];
25        var curValue = parseInt(buyTargetSplit[i].split(":")[1]);
26        buy[curItem] = curValue;
27    }
28    var buildCount = parseInt(splitted[3]);
29    var buildingSplit = splitted[4].split("_");
30    for (var i = 0; i < buildCount; i++) {
31        var curBuilding = buildingSplit[i].split(":");
32        var x = curBuilding[0];
33        var y = curBuilding[1];
34        var team = curBuilding[2];
35        buildings[i] = [x, y, team];
36    }
37    var streetCount = parseInt(splitted[5]);
38    var streetSplit = splitted[6].split("_");
39    for (var i = 0; i < streetCount; i++) {
40        var curStreet = streetSplit[i].split(":");
41        var x = curStreet[0];
42        var y = curStreet[1];
43        var dir = curStreet[2];
44        var team = curStreet[3];
45        streets[i] = [x, y, team];
46    }
47    // TODO: Add game logic here
48
49    printErr("Debug_output");
50    print("ROUNDEND");
51 }

```

Listing 5: PHP code template for the executor

```

1 <?php
2   while(true){
3     $line = readline();
4     $input = explode(';', $line);
5     $cons = array();
6     $items = array();
7     for ($i = 0; $i < 4; $i++)
8       $items[$i] = intval(explode(':', $input[$i])[1]);
9     $buildingCount = intval($input[4]);
10    $buildings = array();
11    for ($i = 0; $i < $buildingCount; $i++) {
12      $buildings[$i] = array();
13      $temp = explode(':', $input[$i + 5]);
14      $buildings[$i]["x"] = $temp[0];
15      $buildings[$i]["y"] = $temp[1];
16      $buildings[$i]["team"] = $temp[2];
17    }
18    $streetCount = intval($input[5 + $buildingCount]);
19    $streets = array();
20    for ($i = 0; $i < $streetCount; $i++) {
21      $streets[$i] = array();
22      $temp = explode(':', $input[$i+6+$buildingCount]);
23      $keyfrom = $temp[0]." ".$temp[1];
24      if ($temp[2] == 'LEFTRIGHT')
25        $keyto = (intval($temp[0]) + 1)." ".$temp[1];
26      else $keyto = $temp[0]." ".(intval($temp[1]) + 1);
27      $cons[$keyfrom][$keyto] = $temp[3];
28      $cons[$keyto][$keyfrom] = $temp[3];
29      $streets[$i]["x"] = $temp[0];
30      $streets[$i]["y"] = $temp[1];
31      $streets[$i]["dir"] = $temp[2];
32      $streets[$i]["team"] = $temp[3];
33    }
34    $fieldCount = intval($input[6+ $buildingCount + $streetCount]);
35    $fields = array();
36    for ($i = 0; $i < $fieldCount; $i++) {
37      $fields[$i] = array();
38      $temp = $input[$i+7+$buildingCount+$streetCount];
39      $temp = explode(':', $temp);
40      $fields[$i]["x"] = $temp[0];
41      $fields[$i]["y"] = $temp[1];
42    }
43    $team = $input[7 + $buildingCount + $streetCount + $fieldCount];
44    $width = $input[8 + $buildingCount + $streetCount + $fieldCount];
45    $height = $input[9 + $buildingCount + $streetCount + $fieldCount];
46    // TODO: Add game logic here
47
48    fwrite(STDERR, "Debug output");
49    echo "ENDROUND\n";
50  }
51 ?>

```