A Creative Learning Sequence in an Introductory Programming MOOC

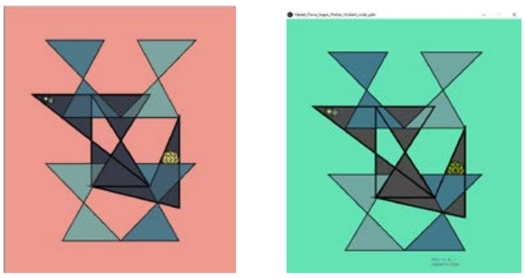
Elisabeth Wetzinger, *elisabeth.wetzinger@tuwien.ac.at*
Gerald Futschek, *gerald.futschek@tuwien.ac.at*
Bernhard Standl, *bernhard.standl@ifs.tuwien.ac.at*
Vienna University of Technology, Austria

Abstract

Lecture-videos, tutorials, and summative examinations frequently give Massive Open Online Courses (MOOCs) the characteristics of teacher-centred instruction. The amount of individual feedback is often reduced to a minimum and most of the practical tasks include multiple-choice quizzes. This is mainly due to the big amount of anonymous and heterogeneous learners. Although a large (massive) number of participants with different goals and pre-education on one hand and compared to that a small group of advisors, on the other hand, suggest simple learning tasks, this situation is not satisfying in respect to our expectation of a constructive and creative learning environment.

In January 2017, we started to design and develop a MOOC for learning computer programming, addressing high-school students without or with little experience in programming intending to study a STEM bachelor programme at our college. Hence, our goal was to conform the heterogeneous levels of pre-knowledge in computer programming of our freshmen students. We decided to use *Processing*[1] as programming language, because it supports an easy learning start to programming by using graphics and animations and allows for an easy transition to Java-like programming languages later on.

As part of the course we designed a learning sequence that addresses the participants' creativity and includes peer feedback, and communication with other participants. We asked students already at a very early stage to create an artwork using previously presented basic graphical procedures. The "artistic" results have to be shown to the other participants combined with some personal remarks. Then they are asked to choose one of the artworks, create a copy of it by coding and provide feedback to the original artist why his or her work was chosen and what challenges they faced in creating a fake copy, optionally completed by further remarks.



Example of an original artwork (left) and its "fake" (right)

In this paper, we present our experience implementing the exercise, results of this learning sequence and selected feedback of our MOOC students concerning the assignment.

Keywords

creative learning sequence; MOOC; learning computer programming;

[1] http://processing.org

Abstract

Since January 2017 we have designed and developed a Massive Open Online Course (MOOC) for learning computer programming, addressing high-school students without or with little experience in programming who intend to start studying at a technical University. The objective of this MOOC is to conform the heterogeneous levels of our freshmen students' pre-knowledge in computer programming. We decided to use *Processing*[2] as programming language because Processing supports an easy learning start to programming by using graphics and animations. We included already at a very early stage of the MOOC a challenging learning sequence that involves creativity, peer feedback and communication with other participants: We asked students to create a graphical artwork through coding using previously in the course presented graphical procedures. Further, we ask them to create a fake copy of an artwork coded by a peer. Finally, artists and fakers should discuss their experiences. This paper summarizes and discusses our experiences with this learning sequence gained during the first run of the MOOC in summer 2017.

Introduction

In the first-year of the bachelor programs of Computer Science and Business Informatics (CSBI), students have a heterogeneous knowledge and experience levels in coding due to different high-school pre-experiences. This fact challenges teaching and studying at the Faculty of Informatics and lead to a high drop-out rate[3]. In order to address these issues, our faculty has set measures to improve the bachelor programme curricula as well as teaching practice. The faculty also has established pre-study bridging courses supporting students at the beginning of their studies in September before the actual lectures start by learning basic skills in programming, maths and other topics of Computer Science. The bridging courses are implemented as on-site workshops of which the programming course is limited to 100 participants following a first-come-first-serve rule. This means, that approximately 80% of students accepted for a CSBI bachelor programme are not able to attend the programming bridging course.

As an additional bridging course for prospective students of the Computer Science and Business Informatics (CSBI) and other STEM undergraduate programs, we developed and provided a massive open online course (MOOC) on introductory programming in German language called "Programmieren mit Processing" (engl.: Programming with Processing). We followed the guidelines published by *Ebner et al* (2014). Especially prospective students, who are not able to attend on-site workshops benefit from the online course as it can be accomplished at one's own pace and can be accessed from anywhere only requiring internet access. As it supports massive numbers of participants, literally every prospective or interested student is guaranteed to be able to accomplish the MOOC.

This MOOC has started in April 2018 at the public MOOC platform iMoox.at[4] and aims at serving everybody who is interested in learning programming using Processing. Our goal is to provide a course that involves constructive learning sequences that provoke the students' creativity, interest, and passion.

In the next sections, we first present the idea of our MOOC and then the description of the learning sequence in detail. We will close our paper with an overview on students' outcomes and conclusions.

[2] http://processing.org (last access: 06/2018)
[3] Source: http://www.tuwien.ac.at (Data from our institution)
4 http://www.imoox.at (last access: 06/2018)

Background

Resnick (2014) claims to give P's a chance: Projects, peers, passion, play. He calls them "The 4 P's of Creative Learning: Projects, Peers, Passion, Play". He argues that active working on meaningful projects supports new ideas in finding solutions. Also, exchange with peers and building on work of others can boost motivation and creative learning processes. Therefore, he also remarks that projects should be rich enough to create passion in finding a good solution. Further, he underlines that a playful way with room for experimentation helps to explore different solutions. Based on that, we are convinced, that learn to code can also be more efficient when these 4 P's of Creative Learning are involved in the instructional design. In order to trigger these so called 4 P's we need not only a suitable programming system but also excellent learning activities. Nevertheless, typical Massive Open Online Courses (MOOCs) rely mostly on passive instructional video learning combined with text-based scripts and multiple choice self-assessment. Koedinger et al (2015) showed that interactive activities support learning more than instructional MOOCs.

For over a decade, computational thinking has been in the focus of educators and researchers in computer science, particularly when it comes to learn how to code. Even though computational thinking has become popular since Wing discussed and reintroduced the topic in 2006 (Wing 2006), computational thinking still has a long tradition in history over decades (Tedre et al 2016). Despite the lack of an overall definition of computational thinking, there is for the most part consensus in including problem-solving involving the thinking skills abstraction, decomposition, algorithmic thinking, evaluation and generalization (Dagiene et al 2017). Hence, learning programming should by its nature support computational thinking skills. At any rate it comprises algorithmic thinking, but while analysing programs and program output also evaluation skills are trained. Since a program is a kind of model of its outcome also abstraction skills may be learned.

Considering both, Resnick's 4P as a framework for building a useful learning environment that conformed to the constructivist approach to coding, and computational thinking as a definition for describing problem-solving in the field of computer science, we developed creative coding tasks as part of our MOOC. In particular, the learning sequence, as presented in this paper allows a playful experimentation while building on another's work and bringing in own ideas leading to a motivating and passionate learning environment. Techniques and skills required in the implementation of the solution are based on the decomposition of existing work, algorithmic reconstruction and the evaluation of the solution.

MOOC: Learning Programming with Processing

The MOOC comprises ten course lectures and is implemented on the virtual learning environment Moodle hosted by our college. The first lecture includes a course welcome and starts the course with a brief overview on fundamental questions about CS and programming, such as "What parts do computers consist of?", "How can we communicate with computers in order to make them solve our problems?", "What are algorithms and abstraction?" and "Why should I learn to program?"

In order to comply with the curricula of the on-site pre-study courses and the Introductory Programming lecture (which is compulsory for all CSBI bachelor students) the remaining course lectures follows a non-objects-first approach, which means, that the object-oriented programming paradigm is not introduced from the beginning explicitly. Course lectures 2 to 10 introduce the Processing Integrated Development Environment (IDE) and basic programming concepts, such as: variables, operators, data types, basic animations and user interactions, decisions using if- and if-else-statements and loops as well as functions, arrays and recursions.

Learning Sequence: Creating Visual Art with Code

The first practical programming exercise is positioned at the end of lecture 2. This lecture introduces the Processing Programming Integrated Development Environment (Processing IDE) which is used throughout the MOOC to write the Processing source code. Three videos and three handouts guide the students through downloading and installing the IDE as well as using it. In addition, they learn how to “draw” simple geometric objects and text by programming basic Processing commands. These include lines, triangles, rectangles or ellipses of different sizes, orientations, positions and the use of colours and transparencies. Finally, an overview of the Processing Reference is given which the course participants are encouraged to use to look up available Processing commands, functions and operators.

The overall exercise topic is to create graphical artworks through coding and to fake-copy a peer’s artwork. The assignment consists of three parts:

Part 1: Programming an artwork

In the first part of the exercise, the participants are asked to create a new sketch (i.e. Processing program) and program a graphical artwork of their own choice by using their previously acquired Programming skills and exploring the Processing Reference.

To keep the challenge of the second part of the exercise feasible especially for programming beginners, we have set the following constraints for each artwork: The sketch canvas size should not exceed 800 by 800 pixels. Only geometric objects of the types “2D Primitives” and “Typography”, referring to the Processing Reference, are allowed. The artwork should not consist of more than 20 of such objects. However, we did not limit the use of colours, contours and transparencies. As part of the exercise, we guide the participants through saving and executing their programs as well as we support them in creating a screenshot of their visual outcome. The submission consists of two parts:

1. Uploading the respective Processing source code file as well as the screenshot of the sketch window, which displays the artwork upon code execution.
2. Creating a thread in a forum provided for the exercise and uploading only the screenshot (but not the source code file) of the artwork, a fitting artwork title, as well as a short description of oneself (programming skills, motivation to take part in the course, optionally intended study programme, or similar).

Part 2: Faking a peer’s artwork through coding

After students have programmed and uploaded their artwork to the forum, every participant is asked to browse through the forum and choose one artwork programmed by a peer. Based on the provided screenshot he or she has to fake the artwork by programming it with Processing. The goal is to create a replica which is as close to the original as possible. The result, both the source code as well as a screenshot of the artwork copy, should be uploaded as a reply to the original forum entry. Along with it they are assigned to describe the main challenges they faced while copying the artwork and encouraged to respond to the personal introduction of the peer by stating similarities or differences concerning motivation for course participation, programming skill, study programme or similar.

Part 3: Discussion

Every participant is encouraged to check his or her own forum thread regularly for replies (i.e. artwork fakes programmed by peers). If a reply is received, the student should compare the fake to the original artwork and explain in the forum how the copy can be identified as such. Also, the original author is supposed to check the fake source code, compare it to the original code and discuss how the Processing programs differ and whether as well as how this is apparent in the resulting visual output or not.

Exercise Grading

Our main objective behind the exercise is to follow the four P’s by Resnick, to foster a creative, playful and individual experimentation with coding and to overcome potential fears or doubts of programming easily. Therefore, we set only the first part of the exercise as compulsory. This means, for the overall course accomplishment we took into account if a student has uploaded an artwork as well as created a forum topic.

The second and third parts are voluntary yet strongly recommended, but not taken into account for grading or a successful course accomplishment. The reason for this decision are the students’ heterogeneous pre-course programming skills which result in various levels of individual challenge and time needed to accomplish the assignments. Our priority was to challenge both beginners as well as advanced students at their individual levels. They should have fun and be able to solve the exercise to their own abilities, rather than leave anyone feeling overloaded or even unable to solve the exercise especially at this early stage of the course.

Didactic Background

With this exercise, we aim at encouraging the students to experiment with code and to take their first steps in programming in a playful, creative and interactive way while practicing the use of the Processing language and the use of the IDE. Without being much aware of the underlying programming concepts or processes the participants should explore and experiment with the use of functions, parameters and the computer graphics coordinate system as well as with the order of code execution. The graphical focus of the exercise and the immediate visual output of their programmed code enables visible feedback of the impact of their programmed code to the output sketch immediately. This supports them in reflecting their written code and the corresponding graphical results through learning by doing.

As the assignment is implemented as the first hands-on programming exercise in the whole course, programming skills are explicitly not needed as a prerequisite. Participants are provided only with basic tools they need to solve the exercise and are encouraged to explore and *“play”* creatively and individually using these tools as a starting point in order to create visual art.

Through this playful hands-on experience, we foster the students’ curiosity and support them in experimenting with coding independently. With the programming reference, they are also equipped to try out further commands and extend their programming skills already at this early stage of the course. Using this approach potential fears and barriers concerning programming and technology can be overcome easily and stress as well as pressure to perform or deliver are avoided. Students are supported to study at their individual level, i.e. by creating simpler or more sophisticated and detailed artworks depending on their own programming experience.

Finally, this exercise aims at community-building among the course participants, i.e. prospective CSBI students already before they start their study programs. We consider this as important as our university is located in downtown Vienna and its distributed building structure does not support informal student socializing well. This means, that in addition to the challenges freshmen face with regard to administrative and organizational aspects as well as the contents of their studies, they also have to start networking with their peers, find friends and build study groups on their own at semester start. With the interactions encouraged between peers especially during this exercise, we support prospective students in getting to know each other prior their studies start so they might be able to master the challenging first weeks of their studies in teams rather on their own.

Results

During the summer holiday break in 2017 we pilot-tested the course with prospective students of the CSBI bachelor programmes. From a total of 315 active course participants, 126 students (i.e. 40%) submitted an artwork (exercise part 1). *Active* refers to having at least one compulsory course activity accomplished successfully. Figure 1 shows examples of artworks programmed and uploaded by MOOC students.

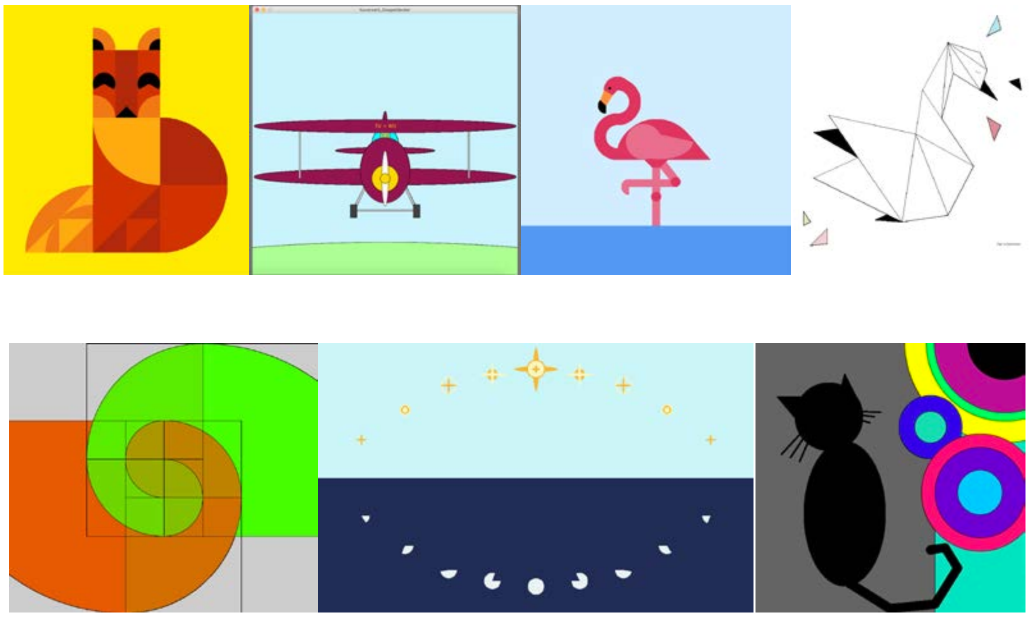


Figure 1. Examples of artworks programmed by course participants (ref. exercise part 1)

These examples visualize the huge variety and level of creativity of the resulting artworks programmed by the course participants using simple Processing commands only.

In total 38 artwork fakes were created by the students. Despite the fact that not every artwork was faked, the resulting copies are incredibly close to the original as Figure 2 shows exemplarily. Based on the results, it can be assumed that the course participants took high efforts in copying the original artworks as exact as possible. Hence, only subtle differences are perceivable, such as in different colour or transparency nuances or command order i.e. occlusion (e.g. artworks in Figure 2, rows 2 and 5) or small differences in position and scale of objects (e.g. artworks in Figure 2, rows 1, 3 and 4).

Participants' Feedback

At the end of each course lecture we asked the students for a brief feedback about what they liked and disliked about it and how they would grade the lecture. In addition, we conducted an anonymous online survey at the end of the course in order to get feedback about the overall course. With respect to the assignment the received feedback shows that the assignment was well-accepted among the students.

"The exercise was very much fun…"

"The exercises were cool. I personally find the interaction with other course participants through a discussion forum a really very good idea. By analysing the others' contributions one gets to know new ways of thinking and becomes more familiar with the geometric objects to be used in the exercise in general. The more participants in the course, the better the result!"

On the one hand, two students were concerned that their artworks should be shared in a forum, on the other hand there were also requests for more exercise of this style throughout the course, because it supports creativity and requires interaction with peers.

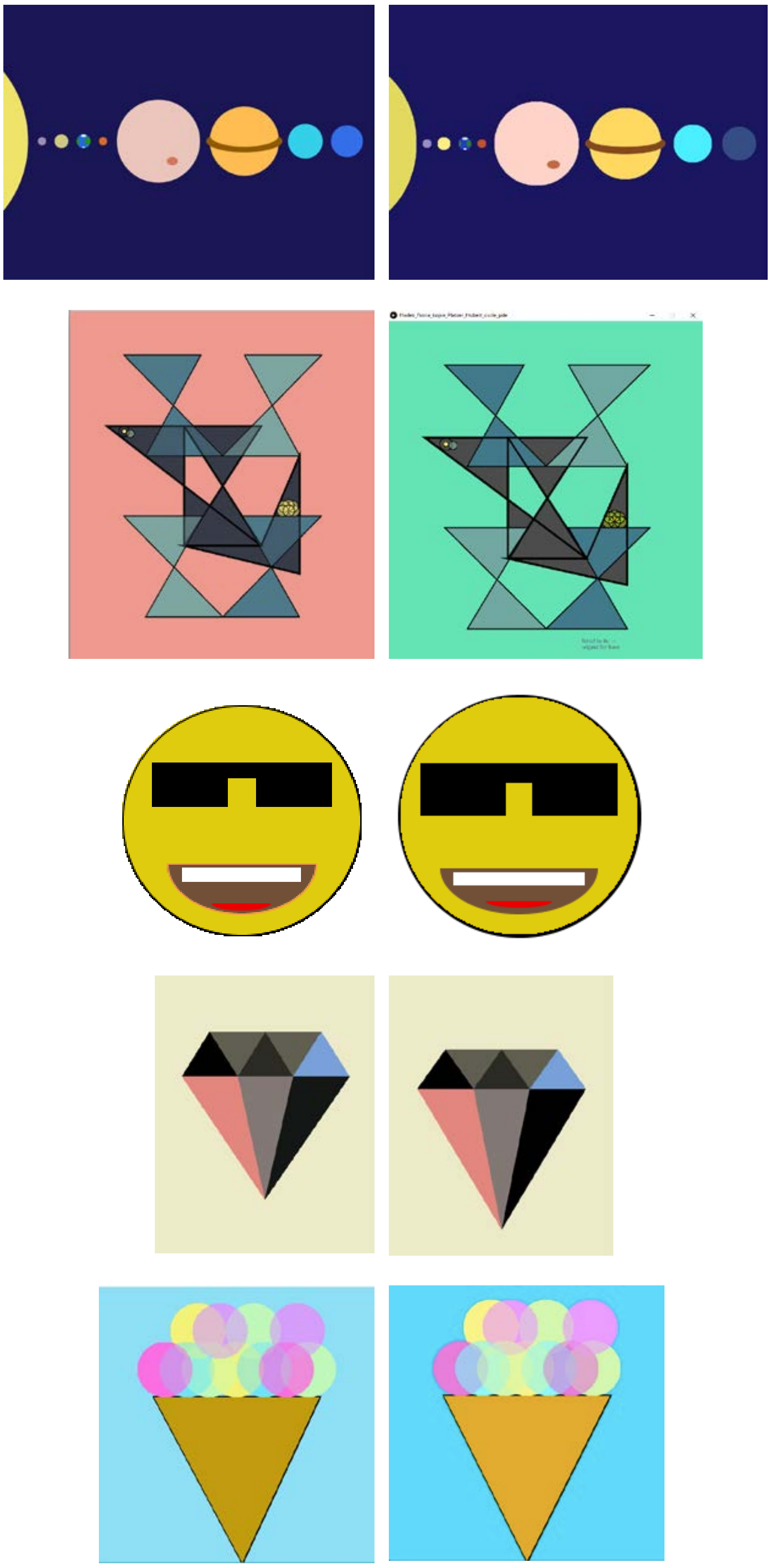


Figure 2. Examples of original artworks (left) and their fakes (right) programmed by course participants (ref. exercise parts 1 and 2)

In several comments, they appreciated that the artwork exercise allowed them to apply the learned content in a creative task and that they could independently create and program their own artwork as well as having the opportunity to choose and be creative in copying artworks of peers. However, one student found it "*very challenging to create an exact copy of the original artwork*". The *"simple introduction to programming allowing for visible feeling of success immediately*" was appreciated and another course participant evaluated the exercise as *"playful yet professional introduction to programming with Processing".*

We compared the feedback on this learning sequence with feedback received on other parts of the course and conclude that this exercise was experienced as very exciting and encouraging.

Conclusion and Discussion

Although we intended to create a task that allows creativity, communication and peer feedback, we were overwhelmed by the passion and playful way of learning by the participants. It is a further proof that the 4 P's of Creativity are powerful. It also shows that creative tasks can enrich a mainly instructive MOOC. The only disadvantage was the relatively low completion rate of the fakes compared to the original artworks observed in our first run of the MOOC. In our opinion task part 2 provokes less creativity, has a higher difficulty and needs a high effort to complete, although it boosts learning evaluation CT skills. In the next runs of the MOOC we will improve this learning sequence according to our gained experiences and we will include further learning sessions that support creativity.

References

Dagienė, V., Sentance, S., & Stupurienė, G. (2017). Developing a two-dimensional categorization system for educational tasks in informatics. *Informatica*, *28*(1), 23-44.

Ebner, M., Lackner, E., & Kopp, M. (2014, October). How to MOOC? - A pedagogical guideline for practitioners. In *The International Scientific Conference eLearning and Software for Education* (Vol. 4, p. 215). " Carol I" National Defence University.

Koedinger, K. R., Kim, J., Jia, J. Z., McLaughlin, E. A., & Bier, N. L. (2015, March). Learning is not a spectator sport: Doing is better than watching for learning from a MOOC. In *Proceedings of the second (2015) ACM conference on learning@ scale* (pp. 111-120). ACM.

Resnick, M. (2014, August). Give P's a chance: Projects, peers, passion, play. In *Constructionism and creativity: Proceedings of the Third International Constructionism Conference*. Austrian Computer Society, Vienna (pp. 13-20).

Tedre, M., & Denning, P. J. (2016, November). The long quest for computational thinking. In *Proceedings of the 16th Koli Calling International Conference on Computing Education Research* (pp. 120-129). ACM.

Wing, J. M. (2006). Computational thinking. Communications of the ACM, 49(3), 33-35.