Computer Animation as a Vehicle for Teaching Computational Thinking

Leonel Morales Díaz[1(✉)] and Laura S. Gaytán-Lugo[2]

[1] Universidad Francisco Marroquín, 6ta Calle Final Zona 10, Guatemala 01010, Guatemala
litomd@ufm.edu
[2] Universidad de Colima, Km. 9 Carretera Colima – Coquimatlán, 28300 Colima, Mexico

Abstract. Several platforms and programming languages exist nowadays designed and built to help educators introduce kids and youngsters into computational thinking. Some of them employ visual elements as the primary output of code in order to provide an immediate and engaging feedback for students. Computer animations, digital drawings and videogames are common products in these environments. With the rising popularity of animated films, children and teenagers may find more attractive to enroll in computer animation courses than in computer programming ones. Based in our own experience conducting computer animation workshops, we believe that this interest can be combined with the aforementioned introductory programming environments to introduce students to both computational thinking and computer animation as complementary subjects. In this paper we will present a general strategy to accomplish this based on what we call animation patterns.

Keywords: Computational thinking · Computer animation · Programming languages · Animation patterns

1 Introduction

Introductory programming languages often categorized as first programming environments, like Scratch [11], Alice [3] and others, can be used to produce moving images and animated stories that range in quality from the rudimentary to highly complex and elaborated. In fact, children, teenagers and adults enrolled in courses that use these environments end up creating animations even if the purpose of the course was to merely teach them how to program.

Producing digital animated films is a task that requires, among others, skills in graphic design and computer programming among others [15, 16]. Several concepts and techniques applied in computer animation are employed by users of first programming environments without realizing that fact. Duration of movements, coordination of several acting agents, change of scenes, interactions between characters, appearance modification, simultaneity, parallel programming, camera movements and several others appear naturally without fancy names in environments like Scratch, Alice, Kodu Game Lab [14], Stencyl [18], and Snap [20].

F.J. Mata and A. Pont (Eds.): WITFOR 2016, IFIP AICT 481, pp. 53–59, 2016.
DOI: 10.1007/978-3-319-44447-5_6

If attendants to computer programming courses that use those environments knew in advance that they would be creating animations and that the skills they learn would actually be of use in a prospective career in computer animation they would feel more motivated to enroll. Knowing only the information the course name provides it is difficult for them to infer that. The importance and effects of choosing a meaningful and attractive title has been studied in other contexts as presented in [12].

On the other hand, if the name of a course on computer programing is going to be changed to something like "computer animation" then the teacher has to be sure that they are is actually teaching the subject. For that purpose, a switch in the emphasis has to be made from teaching programming structures to practicing animation patterns.

In the following sections we will discuss in more detail these three ideas: animation patterns instead of programming structures, utility of the approach to acquire skills that are valuable for the animation industry, and attractiveness of animation in contrast to that of "pure" computer programming to young audiences. The research paths that can be followed to develop these approaches will also be portrayed.

The viewpoints presented in this work are rooted in several years of experience conducting computer animation workshops using first programming environments, aimed precisely to introduce students to computational thinking at the same time. A special section describing these experiences is included.

Because computational thinking is regarded as a fundamental skill in the 21st century world [7], instructional approaches that make it more accessible to young audiences are important to explore and include in the research agenda especially in those countries and regions where the positive impact of technology is expected to be the greatest.

2 Animation Patterns vs. Programming Structures

Traditional introductory programming courses are based on syllabi that include explaining algorithms and several programming structures like decision, loops, variables and types, procedures, definition, etc., that are common to many programming languages [1]. Examples of use are presented with each structure hoping that students will understand the general idea behind each one. First programming environments can be used to teach programming in that way [13, 17] but also to teach computer animation [21].

If the emphasis is to be switched to computer animation then instead of structures the syllabus could be based on teaching and practicing animation patterns. Each pattern represents a technique to create an animated scene and actually provides an opportunity to employ programming structures, only within the context of animation. The approach does not guarantee that all programming structures will be covered, although there is no reason to think that it is not possible, most likely only a subset of them will be constantly used and practiced.

An important difference regards learning the syntax of a programming language. In the traditional approach the syntax plays an important role. It has to be learned at the same time that programming structures have to be understood burdening the attention of students [4, 19]. In introductory programming environments the syntax is much less demanding or even completely taken care of, like in block programming [17, 21].

2.1 Animation Patterns

An animation pattern involves specific movements and effects that are useful and significant in the context of a story told with computer-generated images. The pattern is abstract and requires programming to be implemented with particular characters or objects. The animation patterns thus provide the context to make the programming relevant and meaningful.

Patterns range from the most basic frame by frame and loop through images, to multi-character scenes with interactions among objects that decide in each step what their next movement will be. To use the patterns in a course they must be previously collected, classified according to difficulty or relatedness, and exemplified in an animation that students can build during a class. The goal is to develop in them the ability to use any pattern with any character or scene whenever is needed, which requires a flexibility of mind, a competence for abstraction, and of course programming skills. A suggestion for pattern classification follows:

- Basic patterns with a single character: movement, turns, flips, bouncing, resizing, sudden appearance, fading, color changing.
- Compound patterns: advancing and jumping, moving and turning, resizing and appearing-disappearing.
- Interactive patterns between two or more characters: conversation, coordinated movement, approaching, contact, dancing, contrasting movement, races, collisions.
- Patterns involving backgrounds: switching background, moving background behind still images, adding elements to background.
- Patterns that use randomness: random movement, random turns, random drawing, random movement over one axe, random resizing.
- Patterns of interaction with the user: responding to clicks, responding to keypresses, basic games.

2.2 Programming Structures Through Animation Patterns

As students practice with patterns they employ repeatedly the programming structures needed for the pattern. Advancing a pattern or innovating on it can lead the student to learn autonomously the new programming structures needed or ask the teacher for guidance. Teachers have to be well versed in how to use the different programming structures available in the environment being used, and also they have to be ready to use the questions students make to introduce those structures [8]. Not being properly prepared for such situations can be detrimental of the educational process [6].

3 Value of Computer Animation Skills

Although some first programming environments are well capable of generating high quality animations and video effects they are not the tool of choice for professionals in the industry. It is highly improbable that a person that masters Alice, Scratch or similar,

could be hired in an animation company only on that basis, or at least common job descriptions in the industry so suggest (confront for example [5]).

On the other hand animation concepts used in professional settings are not that different compared to those that can be learned using first programming environments. Finding if students that already master animations in first programming environments are more prepared to learn specialized animation software and tools would provide a strong argument not only for learning animation with first programming environments but also to hire young apprentices in animation companies provided they already do animations in those environments and will complete instruction inside the company. The proposed animation pattern set is subject to fine-tuning to align with industry requirements [21].

4 Attractiveness of Computer Animation

It is very easy to assume that computer animation would be more attractive for young learners than computer programming. It seems that the concept of computer animation is closer to them thanks to the film industry, video games and the pervasiveness of animation in web pages. Finding if in fact when presented with two options for a course, one framed as computer animation and the other as computer programming, students would prefer the first option is a research challenge by itself.

Fig. 1. Computer animation workshop for middle school students in Guatemala, using Alice.

Offering courses on computer animation may also have an effect on future enrollment in computer programming because a first contact between students and programming environments would already be made. Confirming or rejecting these hypothesis would require long term studies. Nevertheless, even if enrollment is not increased the value of computational thinking instruction is by itself a goal worth seeking especially in less developed regions because an important portion of current and future requirement of every job position is related to that skill [2].

On the other hand we have some experience accumulated after several years offering free computer animation workshops using Scratch or Alice for middle and high school students (Fig. 1) and sometimes also for teachers in which case the pedagogical approach is emphasized.

These workshops have been successful in providing an enjoyable first experience with computer programming through computer animation. In the case of Alice workshops, we have recently started paying special attention to how students engage with the language and found that they do in one of four possible ways – as described in previously published works [9, 10] and shortly described in the following subsections.

4.1 Alice Styles of Use

The styles of use or styles of interaction with Alice are particular and distinct ways in which students engage with the platform after an introductory lesson. They become especially apparent when participants are invited to explore freely the environment and build the animation of their choice. The identified styles are four: instruction follower, scene designer, dialogue storyteller and action animator.

Instruction Followers. Students that prefer to follow instructions and copy the actions and choices of the instructor through all the workshop. They tend to use the same characters and to place them as similarly as possible as the instructor is doing on her screen. When they are told to explore and work freely they keep working on the same animation in order to make it a better copy of the model. Usually all students start the workshop following instructions but instruction followers persevere for most time. In fact they seem to enjoy achieving the perfect copy.

Scene Designers. Some students seem less interested in adding movement to their animation and prefer to use Alice as a kind of artist composition tool to create beautiful scenes. These are scene designers. They enjoy adding elements and decorations, changing colors, rearranging components and moving the point of view of the scene using camera movements.

Dialogue Storytellers. A popular style among girls although not limited to them. Dialogue storytellers enjoy placing characters in the scene and making them chat using the “say” and “think” commands. Their code usually includes long sequences of those instructions. They like to modify the dialogues and watch the resulting story.

Action Animators. Some students seem to have a special ability to understand the programming model in Alice and start using advanced commands to create complex

animations in which the story is not the central element but the intensity of the action. These are the action animators.

4.2 A Reflection on Styles of Use and Computer Animation

The study of the different styles of use is still a work in progress, although it allows a brief reflection: would such styles have become noticeable in a setting in which computer programming is emphasized instead of computer animation? If a computer animation course teaches computational thinking as well as a computer programming course but also adds the benefit of being friendlier with different mindsets of students then the computer animation approach seems to be more beneficial. Once again, this idea deserves a specific research effort.

5 Conclusions

This paper has outlined the general idea of using first programming environments like Alice, Scratch, Kodu Game Lab and others, to teach computer animation instead of computer programming but at the same time introduce many concepts and practice structures that are usually part of a computer programming course. One of the points of the article is that computer animation leads to learn computational thinking at the same time.

Three ideas were discussed: animation patterns in computer animation courses replace programming structures as the component units of the syllabus, the expected value of skills and concepts learned with first animation environments for the computer animation industry, and how the attractive of computer animation courses could be greater than that of pure computer programming courses. Each of these ideas induces questions and opens paths of research.

The different styles of use in the particular case of Alice were briefly explained as a first result after some years offering free computer animation workshops with that environment. A computer animation perspective seems to be better suited to accommodate the spontaneous apparition of those styles leading to a more pleasant experience for a wider audience, but at the present time this remains a hypothesis pending validation.

There could be other approaches to teach computer animation that do not require first programming environments but the high availability and the increasing familiarity with them that schools and teachers have, makes them an option worth considering. Additionally the introduction and exposure to computational thinking that is possible with first programming environments seems to make them a better choice in regard of the requirements of modern world technology and labor market.

References

1. Association for Computing Machinery, IEEE Computer Society: Joint task force on computing curricula. Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science. ACM (2013)
2. Hu, C.: Computational thinking: what it might mean and what we might do about it. In: Proceedings of the 16th Annual Joint Conference on Innovation and Technology in Computer Science Education (ITiCSE 2011), pp. 223–227. ACM, New York (2011)
3. Carnegie Mellon University, Alice.org (2015). http://www.alice.org/. Accessed 6 June 2016
4. Lahtinen, E., Ala-Mutka, K., Järvinen, H.M.: A study of the difficulties of novice programmers. ACM. SIGCSE Bull. **37**(3), 14–18 (2005). ACM
5. https://www.prospects.ac.uk/job-profiles/animator Accessed 6 June 2016
6. Bayón, J.B.: Formación del profesorado con scratch: análisis de la escasa incidencia en el aula. Opción **31**(1), 164–182 (2015)
7. Wing, J.M.: Computational thinking. Commun. ACM **49**(3), 33–35 (2006)
8. Assiter, K., Wiseman, C.: Exploratory learning with Alice: experiences leading a computer science workshop for girl scouts. J. Comput. Sci. Coll. **31**(4), 21–27 (2016)
9. Morales Diaz, L., Gaytan-Lugo, L.S., Fleck, L.: Interaction styles in Alice: notes and observations from computer animation workshops. In: Proceedings of the Latin American Conference on Human Computer Interaction. ACM (2015)
10. Morales Diaz, L., Gaytan-Lugo, L.S., Fleck, L.: Profiling styles of use in Alice: identifying patterns of use by observing participants in workshops with Alice. In: 2015 IEEE Blocks and Beyond Workshop (Blocks and Beyond). IEEE (2015)
11. Lifelong Kindergarten Group at the MIT Media Lab, Scratch - Imagine, Program, Share (2015). https://scratch.mit.edu/. Accessed 6 June 2016
12. Hairston, M., Keene, M.: Successful Writing, 5th edn. Norton, New York (2003)
13. Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., Kafai, Y.: Scratch: programming for all. Commun. ACM **52**(11), 60–67 (2009)
14. Microsoft Research, Kodu Home. http://www.kodugamelab.com/. Accessed 6 June 2016
15. Hegg, R.: The art of programming in computer animation. IEEE Spark **4**, 2–3 (2014)
16. Parent, R.: Computer Animation: Algorithms and Techniques. Newnes, Morgan-Kaufmann, Boston (2012)
17. Flanagan, S.: Introduce programming in a fun, creative way. Tech. Dir. **74**(6), 18–20 (2015)
18. Stencyl, LLC, Stencyl: Make iPhone, iPad, Android & Flash Games without code. http://stencyl.com/. Accessed 6 June 2016
19. Jenkins, T.: On the difficulty of learning to program. In: Proceedings of the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences, vol. 4, pp. 53–58 (2002)
20. University of California at Berkeley, Snap! (2016). http://snap.berkeley.edu/ Accessed 6 June 2016
21. Dann, W., Cooper, S.: Education: Alice 3: concrete to abstract. Commun. ACM **52**(8), 27–29 (2009)