IRIS AperTO

UNIVERSITÀ DEGLI STUDI DI TORINO

AperTO - Archivio Istituzionale Open Access dell'Università di Torino

The Creation of Animated Graphs to Develop Computational Thinking and Support STEM Education

This is the author's manuscript

Original Citation:

Availability:

This version is available http://hdl.handle.net/2318/1743142 since 2024-11-20T13:49:19Z

Publisher:

Springer

Published version:

DOI:10.1007/978-3-030-41258-6_14

Terms of use:

Open Access

Anyone can freely access the full text of works made available as "Open Access". Works made available under a Creative Commons license can be used according to the terms and conditions of said license. Use of all other works requires consent of the right holder (author or publisher) if not exempted from copyright protection by the applicable law.

(Article begins on next page)

24 November 2026

The creation of animated graphs to develop computational thinking and support STEM education

Alice Barana[1[0000-0001-9947-5580]], Alberto Conte[1], Cecilia Fissore[1[0000-0001-8398-265X]], Francesco Floris[1[0000-0003-0856-2422]], Marina Marchisio[2[0000-0003-1007-5404]], Matteo Sacchet[1[0000-0002-5630-0796]]

[1] Department of Mathematics, University of Turin, Italy
{alice.barana, alberto.conte, cecilia.fissore, francesco.floris, matteo.sacchet}@unito.it
[2] Department of Molecular Biotechnology and Health Sciences, University of Turin, Italy
marina.marchisio@unito.it

Abstract. Problem solving and computational thinking are the key competences that all individuals need for professional fulfillment, personal development, active citizenship, social inclusion and employment. In mathematics, during contextualized problem solving using Maple, the differences between these two skills become thinner. A very important feature of Maple for problem solving is the programming of animated graphs: an animation obtained by generalizing a static graph, choosing the parameter to be varied and its interval of variation. The first objective of this research is to analyze the computational thinking processes behind the creation of animated graphs for the resolution of a contextualized problem. To this end, we selected and analyzed some resolutions of problems carried out by fourth-grade students of upper secondary schools in Italy (grade 12). The paper shows some examples in which different processes of computational thinking have emerged, which reflect resolutive strategies and different generalization processes. From the analysis it emerged that all the processes underlying the mental strategies of the computational thinking useful for solving problems are activated in the creation of animated graphs. In the second part of the article we discuss examples of animations created during training activities with secondary school teachers, and how animations can support the learning of scientific concepts. It is very important to train the teachers in this regard, both to understand the processes that the students would activate during the creation of animated graphics and to enrich the theoretical or practical explanations with animated representations.

Keywords: Advanced Computing Environment, Animated Graphs, Computational Thinking, Problem Solving, Animations, STEM Education.

1 Introduction

According to the recommendations of the Council of the European Union of 22 May 2018 [1] among the eight key competences for lifelong learning to achieve progress and success are the mathematical competence, defined as “the ability to develop and apply

mathematical thinking and insight in order to solve a range of problems in everyday situations" and the digital competence that “involves the confident, critical and responsible use of, and engagement with, digital technologies for learning, at work, and for participation in society. It includes information and data literacy, communication and collaboration, media literacy, digital content creation (including programming), safety (including digital well-being and competences related to cybersecurity), intellectual property related questions, problem solving and critical thinking". The eight key competences are linked to the development of skills such as: problem solving, critical thinking, ability to cooperate, creativity and computational thinking. They allow to exploit in real time what has been learned, in order to develop new ideas, new theories and new knowledge [1]. Problem solving is an important aspect of mathematics teaching and learning, it occurs in all mathematical curricula [2]. In recent years the use of digital technologies in problem solving activities has allowed a precious variety of representation and exploration of mathematical tasks [3] extending the ways of thinking about the strategies involved in problem solving [4]. One of the technologies used for problem solving activities is Maple, which allows numerical and symbolic calculations, static and animated graphical representations in 2 and 3 dimensions, writing procedures in simple language, programming and connecting all these different registers representation in a single worksheet using verbal language, too [5]. A very important aspect of Maple for problem solving is the design and programming of animated graphs and interactive components. The user can explore through the variation of a parameter and modify inputs.

The first objective of this research is to analyze the computational thinking at the base of the creation of animated graphs for the resolution of a contextualized problem with Maple. To this end, some examples of resolutions of a contextualized problem carried out by fourth-grade students of upper secondary schools will be analyzed. The second objective is to discuss some examples, arising from training activities with secondary school teachers about the creation of animated graphics, to show how animations can support the teaching of STEM, in particular of Mathematics.

2 Theoretical framework

The expression "computational thinking" was popularized by an article by Jeannette Wing [6] in which she supports the importance of teaching the fundamental concepts of computer science in school, possibly from the first classes. Even today the teaching of Computer Science, although frequently described as the systematic study of computational processes that describe and transform information, is reduced in most cases to the use of computers (consumer Computer Science) [7]. Initially, Wing [6] did not give a precise definition of computational thinking but outlined its main features:

- a way in which human beings solve a problem;
- a base for conceptualizing, not programming, on multiple levels of abstraction;
- a base for ideas, not artifacts;
- an integration between mathematical and engineering thinking;
- a process open to everyone, all over the world.

The search for a precise and uniquely shared operational definition of the expression, which still does not exist, can create confusion on the topic, leading firstly to mistakenly consider computational thinking as a new subject of teaching, conceptually distinct from Computer Science, Lodi et al. [8]. The authors believe that it is more important to use the expression “computational thinking” as a short way to refer to a well-structured concept, the founding nucleus of Informatics. Therefore in [8] they describe computational thinking as a mental process (or more generally a way of thinking) to solve problems (problem solving) and define its constitutive elements: mental strategies, methods, practices and transversal competences. Regarding the mental strategies useful for solving problems, the authors describe the following mental processes:

- algorithmic thinking: approach to the design of an ordered sequence of steps (instructions) to solve a problem;
- logical thinking: reasoning to establish and control facts;
- decomposition of problems: dividing and modularizing a complex problem into simple sub-problems, which can be solved more easily;
- abstraction: getting rid of useless details to focus on relevant ideas;
- pattern recognition: identifying regularities and recurrent patterns in data and problems;
- generalization: use the recognized patterns to foresee or to solve more general problems.

These mental strategies recall in many aspects the phases of a problem solving activity in teaching, for example, of Mathematics: understanding the problem, designing the mathematical model, model resolution and interpretation of the results obtained [9]. What distinguishes computational thinking from problem solving is the conceptual paradigm shift constituted by the transition from solving problems to making problem solving [8]. In fact, the first one does not concern a specific resolution of problems: the formulation of the problem and of the solution must be expressed, mainly through an algorithm in an appropriate language, so that an "information processing agent" (human being or machine) can understand, interpret and execute the instructions provided. In our opinion, this difference thins out when problem solving activities are proposed through technologies and in particular Maple. In this case, starting from mental thinking, the student must choose how to set the solution procedure using the available ways (words, graphs, numerical or symbolic calculations, procedures, cycles, etc.) and at the same time write an algorithm in an appropriate language, so that the software processes the information and returns an output. In this research we focus on the creation of animated graphs that involve the generalization of a static graph by choosing the parameter to be varied and its range of variation.

According to Malara [10], the term "generalization process" includes a series of acts of thought that lead a subject to recognize, by examining individual cases, the occurrence of common characteristic elements; to shift attention from individual cases to the totality of possible cases and to extend the identified common features to this totality. The main actions of this process are pattern recognition, identification and connection of similarities. They lead the student to consider all possibilities instead of a single case, and to extend and adapt the identified model. This definition is not so far from the

mental processes of the computational recognition and generalization we mentioned above. About the process of generalization, the author focuses on a reflection by Dörfler [11] that considers the representation of the process to be crucial through the use of perceptible objects, such as written signs, characteristic elements, steps and results of actions. In this way a procedure is generated that allows a cognitive reconstruction and conceptualization of the process itself. The interaction of students with visual representations, in the form of static or animated images can greatly enhance the learning of scientific concepts otherwise expressed only in verbal or mathematical form. As Landriscina explains [12], when people are engaged in tasks that require understanding and reasoning, they also create "mental models", i.e. representations of a dynamic nature close to a certain situation in the external world that serve to make predictions or simulations. In producing or evaluating a “didactic image” it is therefore important to consider not only to what extent it faithfully represents its object, but also which mental models the student will form by looking at the image and which factors will influence this representation. The didactic potential of images and animations is still largely unexplored, as evidenced, for example, by the purely ornamental images that are still printed in many textbooks, and by the limited use of images generated by students in teaching scientific disciplines and learning assessment [12]. It is therefore important to train teachers to know the processes of computational thinking and to understand how to use graphic visualizations and animations in teaching.

3 Animated graphs created with Maple

The creation of an animated graph with Maple2018[1] can be done through the use of the "animate" command, which creates the animation of a graph in 2 or 3 dimensions when a parameter changes. The use of the command can introduce processes that are activated in the generalization of a static graph by choosing the parameter to be varied and its range of variation. The command uses the following syntax:

```
animate(plotcommand,plotargs,t=a..b,options)        (1)
```

where:
- `plotcommand`: a Maple command or a procedure that returns a 2D or 3D graph;
- `plotargs`: list of arguments of `plotcommand`;
- `t`: name of the animation parameter;
- `a,b`: extremes of the interval in which the parameter varies.

Suppose, for example, that the user wants to create an animation to visualize how the concavity of a parabola with a vertex varies in the origin in the Cartesian plane. First, we create the static graph of a particular parabola, for example $y = 3x^2$, using the following command:

```
plot(3*x^2,x=-15..15,color=blue)
```

[1] https://www.maplesoft.com/

By pressing Enter the user obtains the graph illustrated in Fig. 1.

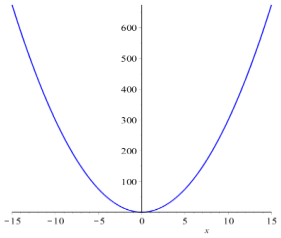


Fig. 1. Static graph of the parabola $y = 3x^2$.

We can do the same process to plot the parabola $y = -3x^2$, drawing a parabola with the concavity facing down. Now suppose we want to visualize through an animation how the concavity of the parabola between the first and the second situation varies. We therefore use the command animate, where, as in (1), the `plotcommand` is `plot` and all the rest into brackets is `plotargs`. In the `animate` command, `plotargs` is not the same as the static graph, but it must be generalized recognizing common patterns between the graphs of the first and second situation that we want to merge into a single command. In particular, in this case we choose the coefficient of x^2 as the parameter to be changed, and the interval in which it varies is represented by the real numbers between 3 and -3. We then get the following command:

```
animate(plot,[a*x^2,x=15..15,color=blue],a=3..-3)
```

In this way we have created the desired animation (Fig. 2) having an immediate feedback of the result of the generalization we have carried out.

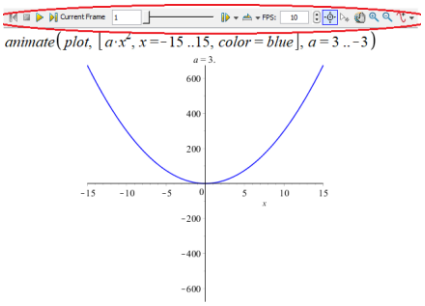


Fig. 2. Animated graph of the parabola $y = a \cdot x^2$, with a varying between 3 and -3.

A further feature of the use of Maple consists in the possibility to define new commands, called "procedures", through a specific programming language. Inside the animate command it is possible to generalize any procedure that outputs a graph in two or three dimensions. It is also possible to export the animated graphic obtained in GIF format to be able to use it as teaching material on its own, to publish it in a Virtual Learning Environment or to reproduce it on any web page [13,14].

In light of the studied theoretical framework, we believe that the creation of the animation of a graph using Maple is a process of computational thinking and the immediate visualization as output of the result can strongly help the students in the development of this competence. We also believe that the use of animations can greatly support the teaching of STEM disciplines. In the following paragraphs we will analyze some examples to demonstrate both aspects.

4 Methodology for the analysis of processes of computational thinking in the creation of animated graphs

To analyze the processes of computational thinking behind the creation of animated graphs, we analyzed the resolutions, using Maple, of a contextualized problem performed by grade 12th students (corresponding to the fourth-grade of upper secondary school in Italy) of different Italian regions in the frame of the Digital Math Training project [15]. The problem, entitled "Ladybug", deals with a ladybug resting on the rear wheel of a bicycle, on the part of the rim closest to the ground. Thinking of the wheel of the bike as a circumference centered in the origin, the position of the ladybug corresponds to the point of tangency of the wheel with the ground, that is the angle of amplitude $\frac{3}{2}\pi$. The total diameter of the wheels (including the inner tube) is 60 centimeters, while the inner tube is 4 centimeters thick. The request of the problem we focused on is the ladybug trajectory in a kilometer, supposing that it has never moved. This trajectory is given by the combination of two motions: the circular one of the wheels and the straight one of the bike. It is therefore represented by a cycloid, whose graph can certainly be considered a correct answer to the request of the problem. However, the static graph does not show how that result was achieved nor the process carried out to arrive at the solution. Instead, through an animated graph it is possible to visualize the motion of the wheel and the path traveled by the ladybug during the motion itself. Moreover, it is possible to keep track of the frames of the animation and in this way the trajectory is directly traced. For our research we have shown 80 solutions proposed by as many students, among which we have selected and analyzed the 16 in which an animation was used to visualize the trajectory.

5 Results

In all the 16 analyzed solutions in which an animation was used, we came across the use of algorithmic thinking to construct the animated graph as a result of an ordered sequence of commands. However, different processes of computational thinking have emerged, some of them successful, reflecting different resolutive strategies. We have classified the analyzed solutions into five categories:

1. animation as creation of a curve point by point (2 solutions);
2. animation as verification of the results (3 solutions);

3. animation as a union of animations (5 solutions);
4. animation of a procedure (1 solution);
5. animations with incorrect procedures (5 solutions).

Although the solutions classified in the same category are not identical, they show analogous processes of computational thinking (as depicted in the theoretical framework). We are going to present an example of a paradigmatic resolution for each category mentioned above. In the solutions of the first category, the graph of the cycloid was obtained through the command `animatecurve` which creates the animation of the trajectory of a curve in 2 dimensions, with the parametric equations of the curve as input. The syntax used in an example of this category was the following:

```
animatecurve([30*sin(t)+30*t,30+30*cos(t),t=0..10*Pi],
color=black,frames=100,thickness=2)
```

We obtained as output a graph drawing the curve point by point:

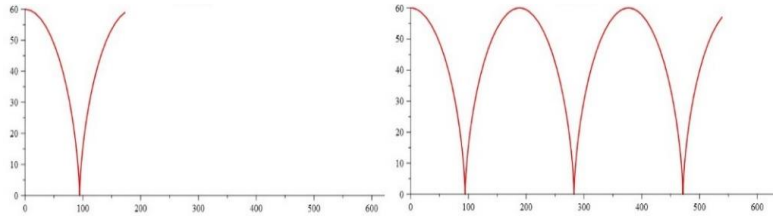


Fig. 3. Example of animation as creation of a curve point by point.

In order to derive the parametric equations of the curve, the student who presented this resolution studied the motions of translation and rotation of the wheel, using the mental process of the decomposition of problems. Then the student used the process of generalization and abstraction to combine the two motions and obtain, with the animate command, the trajectory of the ladybug by calculating its punctual position corresponding to time t, the parameter of the animation. The generalization and use of the command in this case was carried out correctly, even though the solution is not correct because of the wrong starting point.

In the second category we included the solutions in which the student drew the static graph of the cycloid and then animated a point moving on it. In this case, the graph of the cycloid was obtained using the parametric form of the curve with the following command:

```
graf:=plot([0.3(t-sin(t)),0.3(1-cos(t)),t=0..8*Pi],
color=black)
```

The command used to create the animation is the following:

```
animate(pointplot,[[0.3(t-sin(t)),0.3(1-cos(t))],
color=red, symbol=solidcircle],t=0..8*Pi,background=graf)
```

Fig. 4 illustrates two frames of the animation.

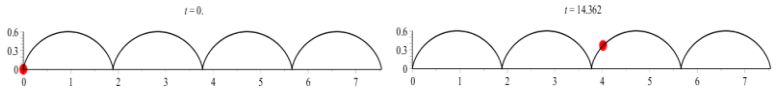


Fig. 4. Example of animation as verification of the results.

This procedure was used to verify the static graph obtained by directly observing how the ladybug moves. In this case the process of generalization concerns exclusively the graph of the point and consequently its coordinates. The graph of the cycloid was used as the background of the animation. The abstraction process took place in two phases: in the first phase a generalization was carried out to create a static command (the parametric curve), in the second phase the generalization was used to create the animation. At the base of the creation of this animation there is the logical process to check the correctness of the previously obtained results.

The third category concerns the solutions in which the animated graph contained: the two circumferences to represent the wheel, the center of the wheel, the ladybug and the radius of the bicycle that joins the ladybug to the center of the wheel. Five different animated graphs were created and then displayed together using the display command. In the following example, the student defined three procedures `coccinel`, `R`, `center` to statically represent the ladybug, the radius and the center of the wheel respectively. Then the student individually animated the graphs of these three elements and of the two circumferences to represent the wheels and finally all the animations were combined in a single graph through the following command:

```
display({animate(R,[t],t=0..8*Pi,scaling=constrained,
frames=80),animate(centro,[3*t,3],t=0..8*Pi,frames=80),
animate(implicitplot,[x^2+y^2+9*t^2-6*t*x-6*y+2.24=0,
x=0..80,y=0..8, color=black], t=0..8*Pi, frames=80),
animate(implicitplot, [9*t^2-6*t*x+x^2+y^2-6*y=0,
x=0..80, y=0..10, color=black], t=0..8*Pi, frames=80),
animate(coccinel,[3*t-2.6*sin(t),3-2.6*cos(t)],t=0..8*Pi,
frames=80,trace=90000)},labels=["x (dm)","y (dm)"])
```

In this command, the trace option allows you to select a set of frames to print while the animation is being performed (Fig. 5). In this resolutive strategy, used by most of the students, the mental process of decomposing problems was used to create animation: a complex animation was divided into different ones, solved in a simpler way. The combination of the animations required the use of the same parameter with the same range of variation, using the process of pattern recognition and generalization.

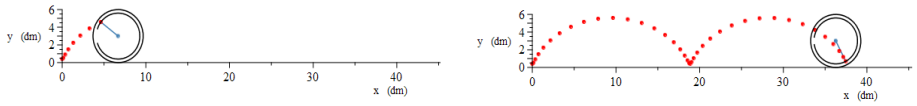


Fig. 5. Example of animation as a union of animations.

The second-last solution analyzed, even if it contains minor accounting errors, is very interesting from the point of view of computational thinking in order to implement it. The ladybug is represented on the left of the graph the ladybug while moving along the wheel and on the right the trajectory is drawn point by point with a segment (Fig 7). Unlike the previous example, the creation of the animation did not require different animations, but different graphical commands with the same parameter were combined in a procedure, which was then animated. The procedure is the following:

```
F:=proc(t)
plots[display](plottools[line]([-2,0],[sin(t)-2,-cos(t)],
color=blue), plottools[line]([sin(t)-2,-cos(t)],
[t,abs(2*sin((1/2)*t))],color=blue),
plot(abs(2*sin((1/2)*x)),x=0..t,color="Green")) end proc
```

This procedure, which requires as input the value of the parameter *t*, returns the static graph of the radius of the wheel, the graph of the curve from 0 to *t* and the segment that joins the position of the ladybug on the wheel to that on the trajectory. For example, the value *t = 3.14* generates the graph shown in Fig. 6.

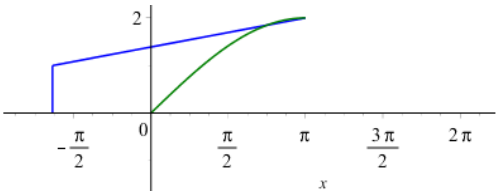


Fig. 6. Static graph for the animation of a procedure.

Then the procedure was animated as the input parameter was changed, adding as a background the static graph of the circumference representing the wheel (Fig. 7). The resolutive strategy is different from the previous one because, instead of joining different animations in a single one, it appears as the creation of a sequence of different snapshots of the same motion. From the point of view of computational thinking this requires a more advanced abstraction process since the generalization through pattern recognition is performed twice: one to define the procedure and one to animate it.

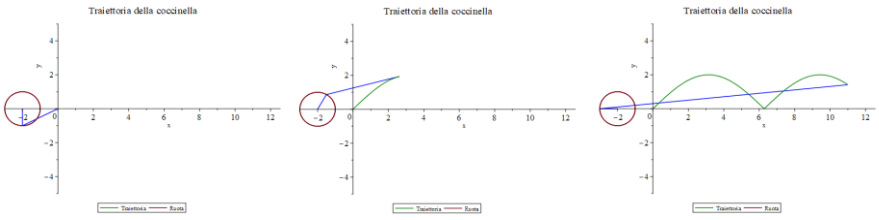


Fig. 7. Example of animation of a procedure.

The remaining five solutions analyzed contain incorrect animations for different aspects. The errors regard the use of the command (incorrect generalization process), the

scale chosen to display the graph (incorrect logical process) or the contextualization of the graph (displayed in the negative semi-axis of the ordinates, arising from a wrong process of abstraction). In the following example (Fig. 8) we can see how the student correctly plotted the graph to illustrate the solution of the problem, to represent the position of the ladybug on the wheel and on the trajectory at the same time. However, already in the static graph, the trajectory of the ladybug on the wheel turns out to be in the negative semi-axis of the ordinates, thus showing an incorrect logical process in the reasoning. In fact, the contextualization of the problem was not considered since the wheel proceeds tangent to the ground level (abscissa axis). The process of generalization was carried out correctly in the creation of the animated graph, but the abstraction process was lacking, it did not check if the solution reflected the context and was explanatory as a proposed solution. This confirms that aspects of computational thinking, as well as of problem solving, are all equally important.

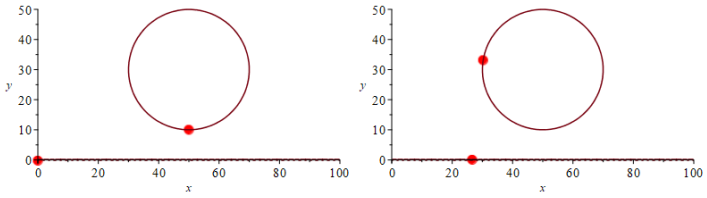


Fig. 8. Example of incorrect animation.

6 Animated graphs for didactics

Since the creation of animated graphs activate all the processes behind the mental strategies of computational thinking, we believe that it can be very important to train the teachers about this, and we add several reasons to support our belief. First, to understand the processes that students will activate when creating animated graphs, for example for problem solving (as shown in the previous paragraphs), but also to illustrate a theoretical concept or build a simulation. Secondly, to enrich the theoretical or practical explanations with animated representations. Thirdly, to enhance the learning of scientific concepts otherwise expressed only in verbal or mathematical form. In the National Project "PP&S - Problem Posing and Solving" [16, 17, 18, 19] during the online training course "Didactics of STEM with Maple" the creation of animated graphs was discussed in various online training meetings. In the case of more elaborate graphs, in which more than one command is used, we explained to teachers how to construct a procedure with a single input parameter, and a graph as output, and then how to animate it (as in the last example of the previous paragraph). During the explanations, the tutor focused the attention of the teachers on the computational thinking processes activated during the creation of animated graphics. In particular, the tutor highlighted the process of pattern recognition and problem decomposition to define a procedure that combines different graphic commands to vary the same parameter and the process of generalization and abstraction to create animation. Below the reader can find some examples divided into two categories: animated problem solving charts, in which the focus is on

training teachers to teach their students how to use Maple for problem solving, and animated graphics for illustration of theoretical concepts.

6.1 Animated graphs for problem solving

The following example concerns a problem on geometric transformations in the Cartesian plane that asks how a craftsman can draw a tile formed by rotated concentric squares (Fig. 9), starting from the outermost square.

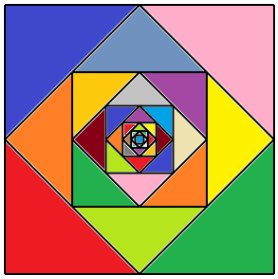


Fig. 9. Tile decoration to be reproduced in the problem.

The solution to the problem is obtained by rotating and expanding the outer square. The static graph of the two transformations is a solution to the problem but it is possible to use animations to illustrate the resolutive procedure step by step. The creation of animated graphs was introduced gradually to the teachers, to underline the process of decomposition of problems, showing how to obtain, in order: the static graph of the dilated and subsequently rotated square; the animation of rotation and expansion; the combination of the two animations to visualize the homothety. With Maple2018, the extension and rotation of a planar figure are obtained respectively with the `dilatation` and `rotation` commands of the `geometry` package. Both commands need three input arguments: the name that the transformed figure must have, the name of the figure to be transformed and the transformation coefficient. To create the animation of the extension and rotation of the square as the expansion coefficient or the angle of rotation varies, the two commands have been generalized defining them as procedures according to the transformation coefficient, indicated with `k`. Below are the commands to define the two procedures (respectively `f` and `g`), where: `quad2` is the starting square, `quadD` is the dilated square and `quadDR` is the rotated dilated square. The commands to display the two animations are also shown:

```
g:=proc (k) options operator, arrow;
plots:-display(draw([quad2(color=blue),(dilatation(quadD,
quad2,k))(color=green)], axes=none)) end proc;
plots:-animate(g,['k'],k=1..(1/2)*sqrt(2));
f:=proc (k) options operator, arrow;
plots:-display(draw([quad2(color=blue),(rotation(quadDR,
quadD,k,'counterclockwise'))(color=green)])) end proc;
```

```
plots:-animate(f,['k'],k=0..-(1/4)*Pi);
```

Multiple strategies can be used to merge the two graphs into one. Since the animations extend over different intervals, teachers proposed to define a new procedure that uses a function that for some times returns the expansion and in another the rotation. In the following commands, the rotation interval was shifted to avoid the pause between the two transformations:

```
gf:=proc (k) options operator, arrow;
piecewise(k<(1/2)*sqrt(2),
plots:-display(draw([quad2(color=blue),(rotation(quadDR2,
quadD2,k-(1/2)*sqrt(2),'counterclockwise'))
(color=green)],axes=none),(1/2)*sqrt(2)<=k,
plots:-display(draw([quad2(color=blue),
(dilatation(quadD2,quad2,k))(color=green)], axes=none))))
end proc;
plots:-animate(gf,['k'],k=1..(1/4)*Pi+(1/2)*sqrt(2))
```

Fig. 10 shows the output of last command.

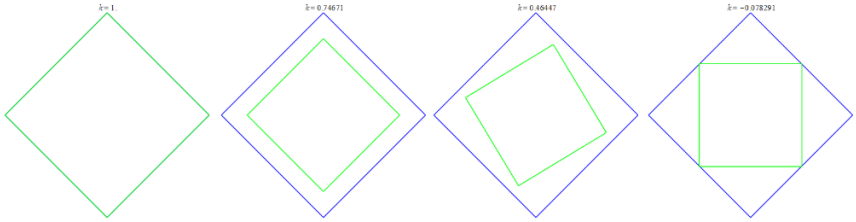


Fig. 10. Animated graph to illustrate the combination of the two transformations.

In this example the process of generalization for the creation of the two animations and the process of abstraction for their union take on importance. This problem gives the chance of multiple insights, addressed together with the teachers, always characterized by the use of animated graphics: development of multiple strategies for the creation of the drawing, iteration of the procedure to create the entire tile and creation of the same type of drawing with other types of regular polygons. In this case, the animations can be used by the teachers to illustrate the solution of the problem but also to introduce or further study the theory on geometric transformations in the Cartesian plane. The animations can be exported in GIF format and loaded into a Virtual Learning Environment where the animation is maintained. In the example in Fig.11, the GIF of the in-depth problem with polygons was attached in a message from the forum of the teacher training course. This type of teaching material can also be used to involve students, to make them curious and passionate about mathematics.

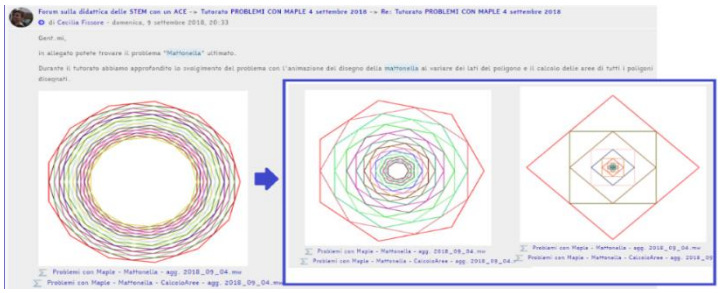


Fig. 11. Example of animation loaded as GIF in a Virtual Learning Environment.

6.2 Animated graphics for illustrating theoretical concepts

A second important aspect in the creation of animated graphs is the use of animations as stand-alone teaching materials to illustrate or enrich the explanation of theoretical concepts or to present simulations, in order to support the construction of mental models by students. The following example concerns the generation of solids of revolution. In addition to a theoretical explanation on the generation of solids of revolution, an example describes how the solid is obtained by rotating a rectangular trapezoid around its major base, minor base, height or oblique side. Graphic visualization is indeed very important for learning because it enriches the theoretical one. In this case, the commands of the `plottools` package are of help, in particular the `polygon` command to draw the trapezoid in the space and the `rotate` command to rotate it by a given angle with respect to a given segment in space. Together with the teachers, a procedure was designed, defined and implemented in several training sessions. Taking as input a figure in space and any segment in space, the procedure returns the animation of the rotation of 360 degrees around the given segment. Following is the procedure:

```
rotaz:=proc (fig,l::list) local gr,rot,fr;
gr:=plots:-display(fig,axes=none);
fr:=proc (k) options operator, arrow;
plottools:-rotate(gr, k, l) end proc;
return plots:-animate(fr,['k'],'k'=0..2*Pi,trace=100,
scaling=constrained, style=pointline,
orientation=[-96, 86, -10]) end proc;
```

As an example, Fig. 12 shows the animation of the trapezoid's rotation around the major side that generates a solid composed of a cylinder surmounted by a cone. Space animations are even more effective because three-dimensional graphs are interactive and have a nice manipulability. It is possible to rotate the figure in each angle, move it and, as in two-dimensional graphics, modify it to study the solid of revolution.

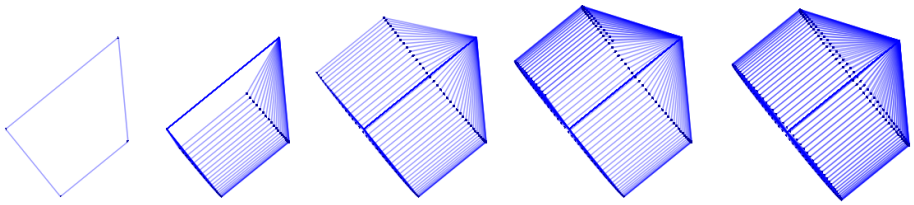


Fig. 12. Three-dimensional animated graph of the generation of a solid of revolution.

In this example, we used the process of decomposition of problems and the process of generalization several times: to move from the representation of a trapezoid in the plane to that in space, to create the trapezoid animation around one of its sides, to generalize the animation to any segment in space and to generalize the animation to any plane figure. The procedure is versatile and allows to create solids of revolution (concave and convex) starting from any shape and segment in the space. The animation in output, exportable in GIF format, is an effective educational tool to illustrate the creation of rotational solids and to support students in understanding their generation, which sometimes can be difficult to imagine. The animations can be a valid aid in the construction of mental models, too, more than a simple static representation of the final result of the rotation or a simply theoretical explanation.

7 Conclusions

The results show some examples of processes of computational thinking in the creation of animated graphs for solving a contextualized problem using Maple. By analyzing the animations created to derive the trajectory of the ladybug, four different processes of computational thinking emerged, reflecting different resolution strategies and generalization processes. From the results, it emerged that the creation of the animated graphs activates all the processes behind the mental strategies of computational thinking, useful for solving problems. It is therefore desirable, because of their additional playful component, to use these graphs in ordinary mathematics education. In the examples we discussed about animations created during training activities with secondary school teachers, we have shown some examples of how animations can support the learning of scientific concepts otherwise only expressed in verbal or mathematical form. We therefore believe that it is very important to train the teachers, both because they need to raise awareness of the processes that the students activate during the creation of animated graphs and to enrich the theoretical or practical explanations with animated representations.

References

1. Council Recommendation of 22 May 2018 on key competences for lifelong learning. Official Journal of the European Union. (2018).

2. Liljedahl, P., Santos-Trigo, M., Malaspina, U., Bruder, R.: Problem solving in mathematics education. Springer Berlin Heidelberg, New York, NY (2016).
3. Santos-Trigo, M., Moreno-Armella, L., Camacho-Machín, M.: Problem solving and the use of digital technologies within the Mathematical Working Space framework. ZDM. 48, 827–842 (2016).
4. Kuzniak, A., Parzysz, B., Vivier, L.: Trajectory of a problem: a study in Teacher Training. 10, 407–440 (2013).
5. Barana, A., Fioravera, M., Marchisio, M.: Developing problem solving competences through the resolution of contextualized problems with an Advanced Computing Environment. In: Proceedings of the 3rd International Conference on Higher Education Advances. Universitat Politècnica València, 1015–1023 (2017).
6. Wing, J.: Computational thinking. Presented at the Communications of the ACM (2006).
7. Bizzarri, G., Forlizzi, L., Proietti, G.: Informatica: didattica possibile e pensiero computazionale. 10 (2011).
8. Lodi, M., Martini, S., Nardelli, E.: Abbiamo davvero bisogno del pensiero computazionale? 15 (2017).
9. Samo, D.D., Darhim, D., Kartasasmita, B.: Culture-Based Contextual Learning to Increase Problem-Solving Ability of First Year University Student. Journal on Mathematics Education. 9, (2017). https://doi.org/10.22342/jme.9.1.4125.81-94.
10. Malara, N.A.: Processi di generalizzazione nell'insegnamento/apprendimento dell'algebra. Annali online formazione docente. 4, 13–35 (2013).
11. Dorfler, W.: Forms and means of generalization in mathematics. In: Mathematical Knowledge: Its growth through teaching, Kluver, pp. 63–95 (1991).
12. Landriscina, F.: Didattica delle immagini: dall'informazione ai modelli mentali. 12, 27–34 (2013).
13. Barana, A., Marchisio, M., Miori, R. MATE-BOOSTER: Design of an e-Learning Course to Boost Mathematical Competence, in: Proceedings of the 11th International Conference on Computer Supported Education (CSEDU 2019), pp. 280–291 (2019).
14. Barana, A., Fioravera, M., Marchisio, M., Rabellino, S.: Adaptive Teaching Supported by ICTs to Reduce the School Failure in the Project "Scuola Dei Compiti," in: Proceedings of 2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC). Presented at the 2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC), IEEE, pp. 432–437 (2017).
15. Barana, A., Marchisio, M.: Dall'esperienza di Digital Mate Training all'attività di Alternanza Scuola Lavoro. MONDO DIGITALE. 15(64), 63-82 (2016).
16. Brancaccio, A., Marchisio, M., Palumbo, C., Pardini, C., Patrucco, A., Zich, R.: Problem Posing and Solving: Strategic Italian Key Action to Enhance Teaching and Learning Mathematics and Informatics in the High School. In: Proceedings of 2015 39th Annual Computer Software and Applications Conference. pp. 845–850. IEEE, Taichung, Taiwan (2015).
17. Demartini, C.G., Bizzarri, G., Cabrini, M., Di Luca, M., Franza, G., Maggi, P., Marchisio, M., Morello, L., Tani, C.: Problem posing (& solving) in the second grade higher secondary school. Mondo Digitale 14, 418–422 (2015).
18. Barana, A., Conte, A., Fioravera, M., Marchisio, M., Rabellino, S.: A Model of Formative Automatic Assessment and Interactive Feedback for STEM, in: Proceedings of 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC), IEEE, Tokyo, Japan, pp. 1016–1025 (2018).
19. Brancaccio, A., Esposito, M., Marchisio, M., Pardini, C.: L'efficacia dell'apprendimento in rete degli immigrati digitali. L'esperienza SMART per le discipline scientifiche. MONDO DIGITALE 15(64) 803-821 (2016).