

Next-Lab

Next Generation Stakeholders and Next Level Ecosystem for Collaborative Science Education with Online Labs

Innovation Action in European Union's 2020 research and innovation programme

Grant Agreement no. 731685

next lab

Deliverable 3.7

Final releases of the ePortfolio and modelling app

Editor	Denis Gillet (EPFL)
Date	31 st December 2019
Dissemination Level	Public
Status	Final



© 2019, Next-Lab consortium

The Next-Lab Consortium

Beneficiary Number	Beneficiary name	Beneficiary short name	Country
1	University Twente	UT	The Netherlands
2	École Polytechnique Fédérale de Lausanne	EPFL	Switzerland
3	IMC Information Multimedia Communication AG	IMC	Germany
4	EUN Partnership AISBL	EUN	Belgium
5	Ellinogermaniki Agogi Scholi Panagea Savva AE	EA	Greece
6	University of Cyprus	UCY	Cyprus
7	Universidad de la Iglesia de Deusto	UD	Spain
8	Tartu Ülikool	UTE	Estonia
9	Núcleo Interactivo de Astronomia Associacao	NUCLIO	Portugal
10	École Normale Supérieure de Lyon	ENS de Lyon	France
11	Turun Yliopisto	UTU	Finland
12	University of Leicester	ULEIC	United Kingdom

Contributors

Name	Institution
Denis Gillet, André Lemos Nogueira, Juan Carlos Farah, Yves Piguët, Joana Soares Machado, Isabelle Vonèche Cardia, and Ezequiel Gonzalez Debada	EPFL
Anjo Anjewierden and Ton de Jong	UT
Internal Reviewers	
Evita Tasiopoulou	EUN
Leo Siiman	UTE

Legal Notices

The information in this document is subject to change without notice.

The Members of the Next-Lab Consortium make no warranty of any kind with regard to this document, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The Members of the Next-Lab Consortium shall not be held liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material.

The information and views set out in this deliverable are those of the author(s) and do not necessarily reflect the official opinion of the European Union. Neither the European Union institutions and bodies nor any person acting on their behalf may be held responsible for the use which may be made of the information contained therein.

Executive Summary

This deliverable primarily presents the final releases of the ePortfolio features of Graasp and the modelling app (GoModel).

The initial specifications of these two components of the Go-Lab ecosystem were defined in September 2017 in D3.1 (*Specifications, mock-ups and/or prototypes of 21st century apps, self- and peer assessment apps, ePortfolio, and modelling app*) and their first releases were presented in June 2018 in D3.4 (*First releases of the ePortfolio and modelling app*).

The final release of the ePortfolio features fully relies on the “Export as an eBook” option of Graasp to comply with the EU General Data Protection Regulation (GDPR), i.e., not to request any personal information from the students and not to ask them to create an account for accessing the Go-Lab ecosystem and the content of their ILSs.

The off-line ILS viewers developed in the framework of the sister H2020 GO-GA project (Go-Lab Goes Africa) also provides an interesting alternative to keep a fully independent and working copy of the ILS content and learning outputs for students.

The final release of the GoModel modelling app is based on experience with the initial release described in D3.4 and improvements to the Go-Lab technical infrastructure which resulted in significant parts of the app to be adjusted and improved.

A selection of apps complementing GoModel and supporting the development of mathematical and computational thinking activities is also presented.

As a “release” document, this report is a description of the software actually implemented, with the current online versions of the Graasp ePortfolio features and the GoModel app available on Golabz representing the actual deliverable.

Table of Contents

1. Introduction	7
2. ePortfolio Features	8
3. Player-based Archiving	10
4. GoModel App	11
4.1 Description	11
4.2 Practical Experience and Improvements	13
4.3 Conclusions	14
5. Code App	15
6. Sysquake App	20
7. GeoGebra App	23
8. Scratch App	25
9. Conclusions and Outlook	27

1. Introduction

This deliverable primarily presents the final releases of the ePortfolio features of Graasp and the modelling app (GoModel).

The initial specifications of these two components of the Go-Lab ecosystem were defined in September 2017 in D3.1 (*Specifications, mock-ups and/or prototypes of 21st century apps, self- and peer assessment apps, ePortfolio, and modelling app*) and their first releases were presented in June 2018 in D3.4 (*First releases of the ePortfolio and modelling app*).

The final release of the ePortfolio features fully relies on the “Export as an eBook” option of Graasp to comply with the EU General Data Protection Regulation (GDPR), i.e. not to request any personal information from the students and not to ask them to create an account for accessing the Go-Lab ecosystem and the content of their ILSs.

The off-line ILS viewers developed in the framework of the sister H2020 GO-GA project (Go-Lab Goes Africa) also provides an interesting alternative to keep a fully independent and working copy of the ILS content and learning outputs for students.

The final release of the GoModel modelling app is based on experience with the initial release described in D3.4 and improvements to the Go-Lab technical infrastructure which resulted in significant parts of the app to be adjusted and improved.

As a “release” document, this report is a description of the software actually implemented, with the current online versions of the Graasp ePortfolio features and the GoModel app available on Golabz representing the actual deliverable.

The two first sections focus on the ePortfolio. Section 2 presents the evolution of the ePortfolio features of Graasp since their first releases, Section 3 presents how the off-line ILS viewer can also be used as ePortfolio.

The next sections focus on apps to support the study of dynamical models. Section 4 presents the evolution of the GoModel app, while Section 5 and 6 present general-purpose alternative apps that can also be used for dynamical simulation of systems modelled by differential equations in particular, and for computational thinking or data science in general. Section 7 summarizes how the GeoGebra app can support exploration of mathematical concepts and functions, while Section 8 tackles graphical programming to further support computational thinking activities, especially with young children.

Finally, Section 7 provide conclusions and outlook.

2. ePortfolio Features

As highlighted in D3.1 and implemented for D3.4, the built-in Graasp export features support the core objectives of an ePortfolio, i.e., providing students with a way to keep copies of their learning outcomes for later use or sharing.

Since their initial releases, the export features have been improved for students. The user interface is still similar but all the backend for the generation of PNG images, PDF files and eBooks, has been rewritten to be more efficient and is running on a dedicated Amazon Web Services (AWS) cloud infrastructure in Europe.

During the last year, investigations were carried out to see how more interaction could be added inside eBooks, such as enabling playing YouTube videos or even interacting with virtual labs. However, and despite it was possible to implement such features, the generated eBooks were not compatible with the standard eBook readers and this could have cause negative user experience. So, it was decided for the end of the project to go back to a static standard eBook implementation. As a matter of fact, this simple implementation is more in line with the “archiving” dimension of ePortfolios.

The export menu accessible by students by clicking on the “Cloud” icon on the top-right corner of the standalone view is shown on Fig. 1. Teachers can also access this menu in the Page view and even create eBooks integrating multiple ILSs (when generated from an embedding space).

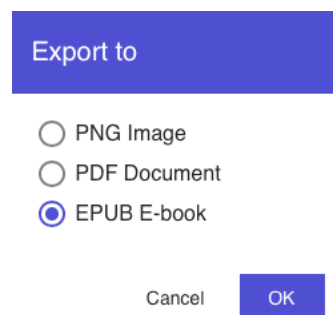


Figure 1. Export menu accessible by students when clicking on the “Cloud” icon on the top-right corner of the standalone view.

As the ILSs integrate external resources and third-party apps and labs, the outputs produced as PNG, PDF or eBooks is often unpredictable. The usage of the apps and labs development framework presented in D2.10 is the best guarantee for quality and well-integrated outputs.

For PDF files added in the various phases of ILSs, all their pages are integrated separately in the PNG, PDF or eBooks to facilitate a posteriori reading.

One should highlight that the outputs produced as ePortfolios integrate the content provided by the teachers (such as descriptions, documents, guidelines and preconfigured apps and labs), the learning outputs created by the students (inputs in apps and labs, comments, notes), as well as activity traces turned into learning analytics if such apps have been added and made visible for students by the teachers. Moreover, any teacher or peer feedback will also be saved in an ePortfolio, thereby allowing students to reflect later on commented versions of their work.

Fig. 2 shows parts of the eBook generated from an ILS integrating an app, a lab, and a learning analytics dashboard. Fig. 2a shows the table of content, Fig. 2b shows the “Investigation” phase with the “Electrical circuit” app as personalized by the student and the “Analytics” phase with the “Timeline” app including the personal traces of the student (Student) and the anonymized traces of the other students, if any (Student 1).

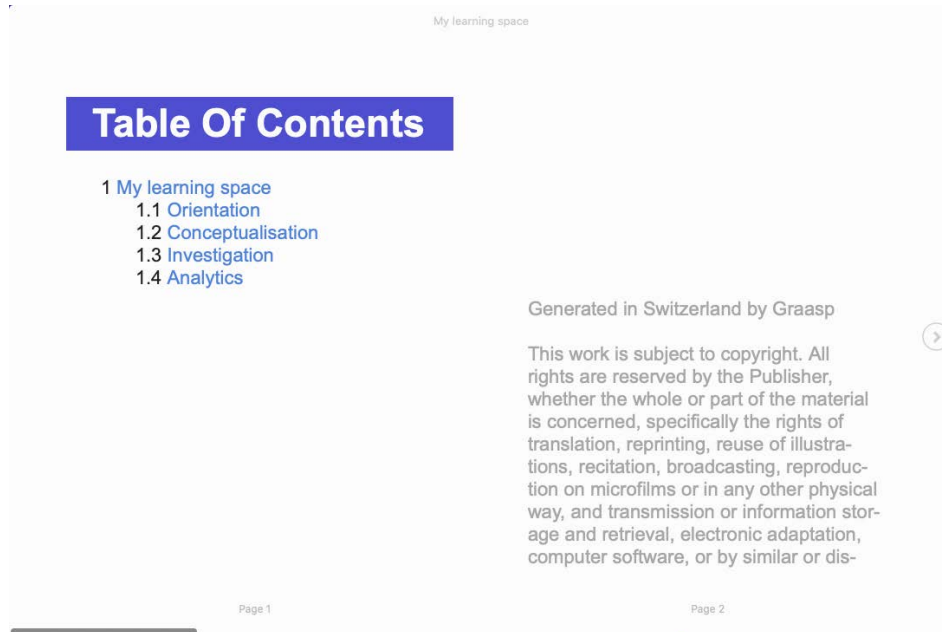


Figure 2a. Part of an eBook generated by a student from the standalone view of his or her ILS (table of content).

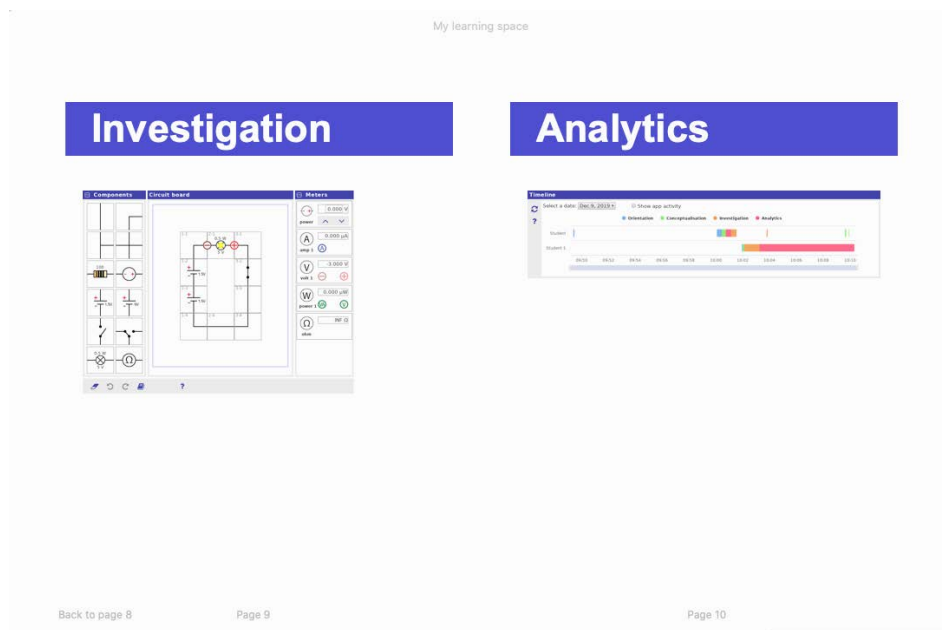


Figure 2b. Part of an eBook generated by a student from the standalone view of his or her ILS (Chapters corresponding to the “Investigation” and “Analytics” phases as named by the teacher).

3. Player-based Archiving

As mentioned in the introduction, the features developed in the framework of the sister H2020 GO-GA project (for off-line use of ILSs in the African context) are providing a very interesting alternative to the export features described in Section 2. This approach has the advantage of avoiding the formatting issues that might be experienced during the conversion to PNG, PDF or eBook formats.

ILS desktop viewers that can be installed on laptop or desktop computers running Windows, Mac, or Linux, have been developed. They can be downloaded from the bottom blue part of the Graasp landing page.

Teachers can export ILSs as archives on memory sticks together with the viewer application and share them with their students (sharing can also be done through the network when available). Once the students load the exported ILSs on their computer and use them, they can save the populated versions again as an archive and “Load” (Fig. 3) them later for restoring, just viewing, or sharing their achievements.

Currently, only teachers can export ILSs as off-line ready archives. This feature will also be made available to the students from the export menu of the standalone view in 2020 in the context of the GO-GA project. To import an ILS in the Graasp ILS desktop viewer, the “Visit a Space” option (Fig. 3) has to be selected and the short link of the standalone view has to be provided. The corresponding ILS has to be made public just during this operation accounting for the fact that the off-line viewer cannot share credentials with Graasp. The ILS has also to be set as off-line ready in its settings (these two constraints will be removed for students). Then, the ILS can be previewed and saved in the viewer. After use, the ILS including the student’s outputs can be exported as an archive on hard drives or memory sticks (using the export button located at the top-right corner of the viewer app).

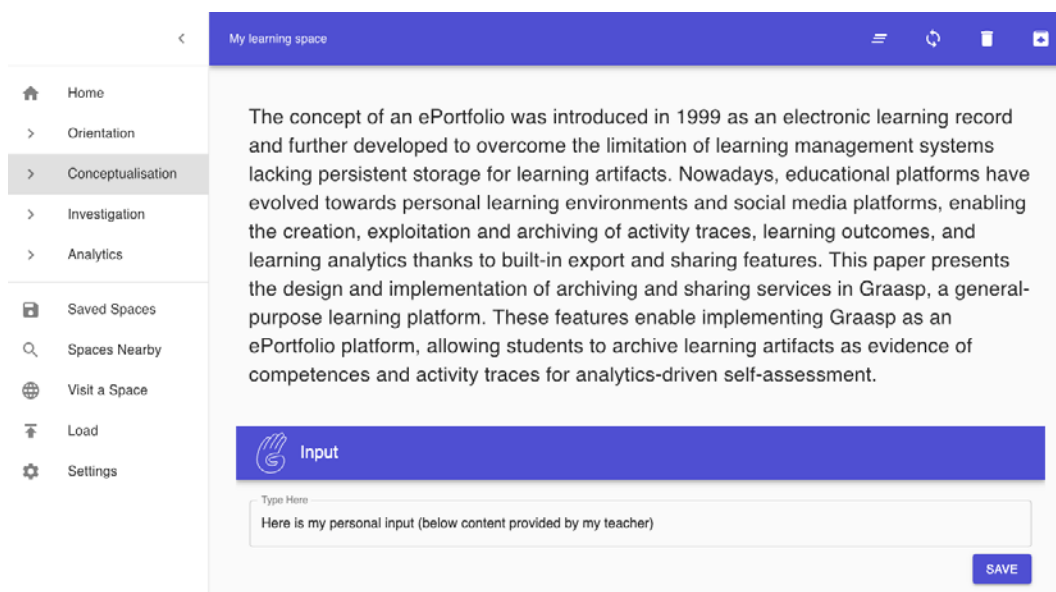


Figure 3. ILS imported in the standalone Graasp ILS desktop viewer and including a personal note of the student (in the input app).

4. GoModel App

4.1 Description

The initial version of GoModel, as described in D3.4 has been used in various ILSs. In this section we reflect on this use and the work done towards the final release.

GoModel provides a graphical interactive interface to model scientific phenomena by relating the variables relevant in the domain with (mathematical) relations. The notation GoModel uses to create models is based on system dynamics in which the relations define how the variables change over time. Once a model has been created it can be mathematically simulated and the results are shown in a graph.

An example to illustrate the modelling approach is population growth. Given a population (P), and a growth rate (r), the growth of the population is defined by the differential equation:

$$dP/dt = \text{growth} = r * P$$

where dP/dt is how P **changes** over time (t).

Fig. 4 contains the population growth model created in GoModel (a description of the UI to achieve this can be found in D3.4). The blue *population* is a stock, whose values can increase (cloud with arrow from the left), or decrease. The auxiliary *growth* defines the increase, and the growth depends on the constant (*rate*) *rate*. Arrows between the graphical elements denote dependencies. The model is completed by filling in the details of all elements, for example:

- population (initial): 1
- rate (constant): 0.1
- growth: rate * population

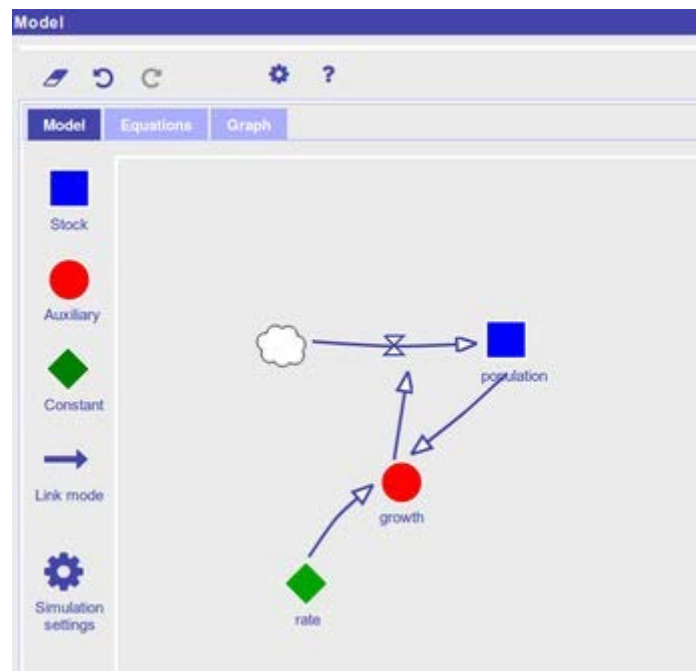


Figure 4. Population growth model in GoModel.

The model is now ready to be simulated. In this particular model the outcome over time, what happens to the population in the future, is simple to predict: exponential growth.

One way, GoModel is being used is to provide students with an initial model and ask whether changing the values of a constant changes the model's behavior over time in a fundamental way, or what the role of a particular constant is. We illustrate this with the following model. This is the same basic model as above, with an added mystery variable called K. The growth over time is given as:

$$dP/dt = \text{growth} = r * P * (1 - P/K)$$

When a student observes the behavior of this model over time with $r = 0.1$, $K = 100$ and an initial P of 1, then the outcome is like that in the Fig. 5.



Figure 5. Evolution of a population.

The graph suggests that this model also exhibits exponential growth. Modifying the model slightly, by setting $r = 0.2$ for instance, the student might discover that this is not necessarily the case (see Fig. 6).

On the left, the blue line represents the value of the population, whereas on the right the red line is the value of growth contributing to the population over time. The student could correctly conclude that in this model, the population growth decreases when the population reaches a critical maximum value. Further experimentation will reveal that this maximum is always equal to the mystery variable K (= 100 in the examples).

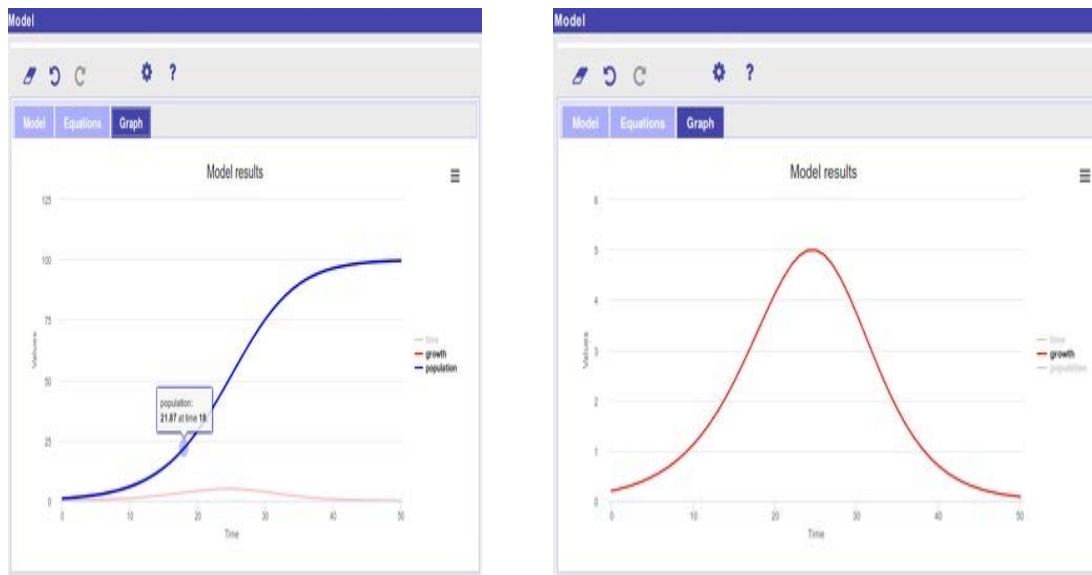


Figure 6. Alternative evolutions of a population depending on the model.

4.2 Practical Experience and Improvements

After the initial release of GoModel the underlying tool development infrastructure has been improved significantly. This provided an opportunity to align the user interface and other services of the infrastructure of GoModel with other tools and apps. Unfortunately, GoModel was written in CoffeeScript and based on legacy code, and the transformation to the new infrastructure required a major effort, essentially reimplementing a significant portion of the app.

During the transition to the new implementation we considered issues and suggestions by teachers based on their experience. Below is a brief overview.

A general point mentioned by teachers is to allow students to review previous versions of their models. This is not easy to deal with given that in the infrastructure there is a one-to-one correspondence between an app and the underlying model. This also applies to, for example, the concept mapper. One approach is to include multiple copies of GoModel in the ILS, and let the student switch between these to review an earlier version.

It frequently happened that students created a model graphically and entered the descriptive names of stocks and constants. Only later they discovered that *рост* or *աճ* (growth in Cyrillic or Armenian) and *Население* or *բնակչություն* (population) were not recognized when entered into the formula of an auxiliary. The reason for this is that the formula is passed directly to the mathematics parser used in GoModel. And this parser assumes variable names to use Latin. As consolation, other mathematics parsers such as Wolfram alpha, also fail to deal with non-Latin alphabets. Allowing arbitrary descriptive names to be entered by students has been addressed by internally associating a Latin symbol with each of the variables and replacing the name entered by the student with this symbol before calling the mathematics parser.

Most of the other issues encountered are related to entering mathematics for the auxiliaries. When the formula could not be parsed the UI showed a red box around the formula, and the student had to figure out the cause. Analysis reveals that there are four plausible causes for a parse error, and by separating these out it becomes easier to point to the cause.

1. Name not recognized. This has been described and solved above.
2. Student wants to use a function, for example square root, and does not know how to specify it. This has been solved by adding a menu with the available functions.
3. A variable is used but not part of the model yet. This is not necessarily wrong, and a warning is given, naming the variable(s).
4. There is a genuine error in the formula. This is for the student to solve.

4.3 Conclusions

The most exciting aspect of GoModel is that students can easily **create** a model, **run it**, and **explore** the outcome in a graph (which by itself is interactive). The ability to be able to create a model makes it very distinct from the simulations in Go-Lab in which only values of variables can be manipulated. And the ability of running the model makes it very distinct from apps such as the concept mapper.

GoModel falls into a family of modelling tools that are now technologically feasible and are increasingly becoming more widespread. An example is Scratch (<https://scratch.mit.edu/>) for computational thinking (See Section 8).

A possible next step is to remove the system dynamics constraint that all relations are based on time. On one hand, the time constraint limits the types of models that can be created, and on the other hand it keeps the graphical representation of the model relatively simple. Removing the time constraint, therefore has implications for the graphical representation. An interesting challenge to address.

5. Code App

As mention in Section 4, the GoModel app is a contribution towards supporting Computational Thinking approaches in education. To further strengthen this dimension and to cope with requests from STEM teachers that have been asked to also cover the new programming and data science elements of 21st-century secondary school curricula, we are developing a Code app supporting programming in Python and JavaScript. Python is typically the programming language chosen for developing digital skills directly from the beginning of the instruction or once the limits of graphical programming environments like Scratch are reached. Python is also progressively replacing C and C++ in bachelor studies. JavaScript being the language built in browsers and the choice for designing attractive user interfaces, it is also an interesting alternative.

The acquisition of Computational Thinking skills relying on Python is largely supported in universities by using the Jupyter Notebook framework (<https://jupyter.org>). This framework however requires heavy virtual machine infrastructures that need to be installed and exploited by the educational institutions. Providing a cloud alternative to such environments as part of the Go-Lab ecosystem is a great opportunity for schools not having IT services to offer Python and for the Go-Lab initiative to become even more visible.

When installed in an ILS from the “Select app from Graasp” popup, the Code app can be configured by the teacher in the authoring view as show in Fig. 7.

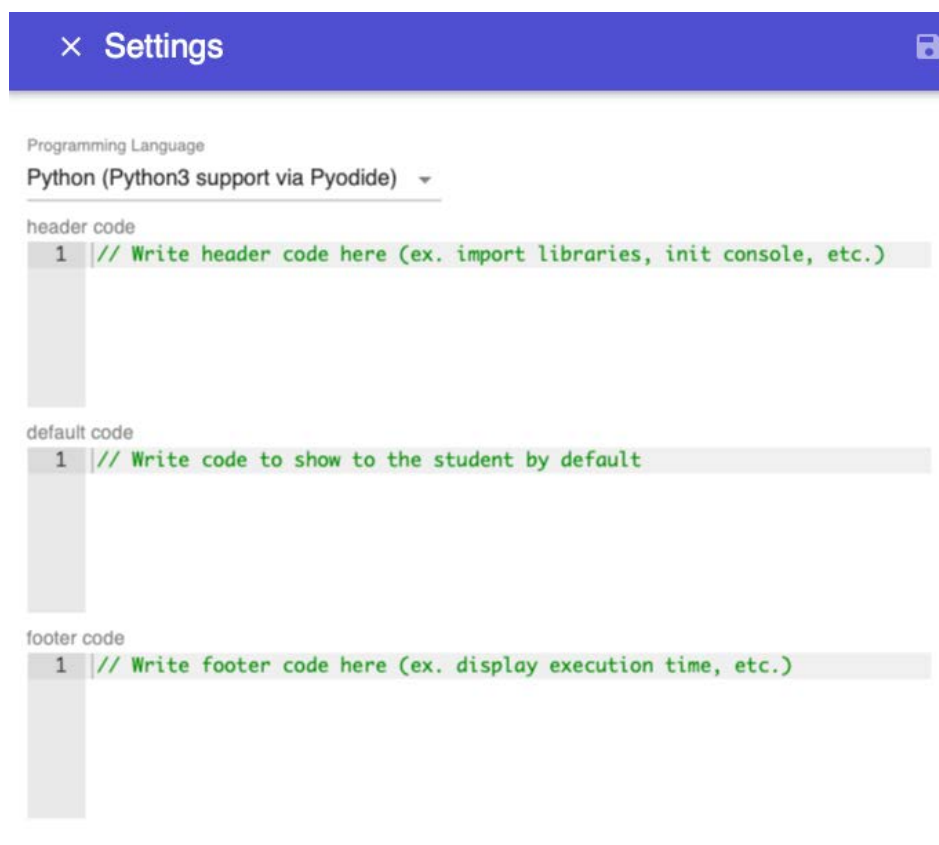


Figure 7. The teacher configuration panel of the Code app.

The popup menu on the top of the settings view enables to choose between JavaScript and Python. Typically, one Code app is added for each code snippets that the teacher would like to share with his or her students or for each exercise proposed to the students. The code is split in three parts. Only the second part is visible for the students. The first one includes code to be executed for setting up the environment, like importing required libraries. The third part includes code to be executed at the end, like displaying the execution time. In the second part for the students, some hints or partial code can be provided.

Fig. 8 show the example of code added by a teacher to let the students test if a number is even or not. Fig. 9 show the standalone view with the code modified by a student and its outputs when executed in the black console. The button with a white arrow in the blue circle triggers code execution when pushed and the button with a floppy disk icon enables the student to save his or her code.

```
default code
1 x = 1
2 if x % 2 == 0:
3     print("Congratulations!")
4 else:
5     print(x, " is not an even number")
```

Figure 8. Code snippet predefined by the teacher.

The Computational Thinking activity template has been chosen for the ILS below instead of the basic inquiry learning scenario (see D2.10 for selecting alternative pedagogical scenarios in Graasp).



Figure 9. Code snippet edited and executed by the student in the standalone view.

The teacher can see the contributions of all students and provide inline comments to them in the Graasp authoring view as shown in Fig. 10 and 11.

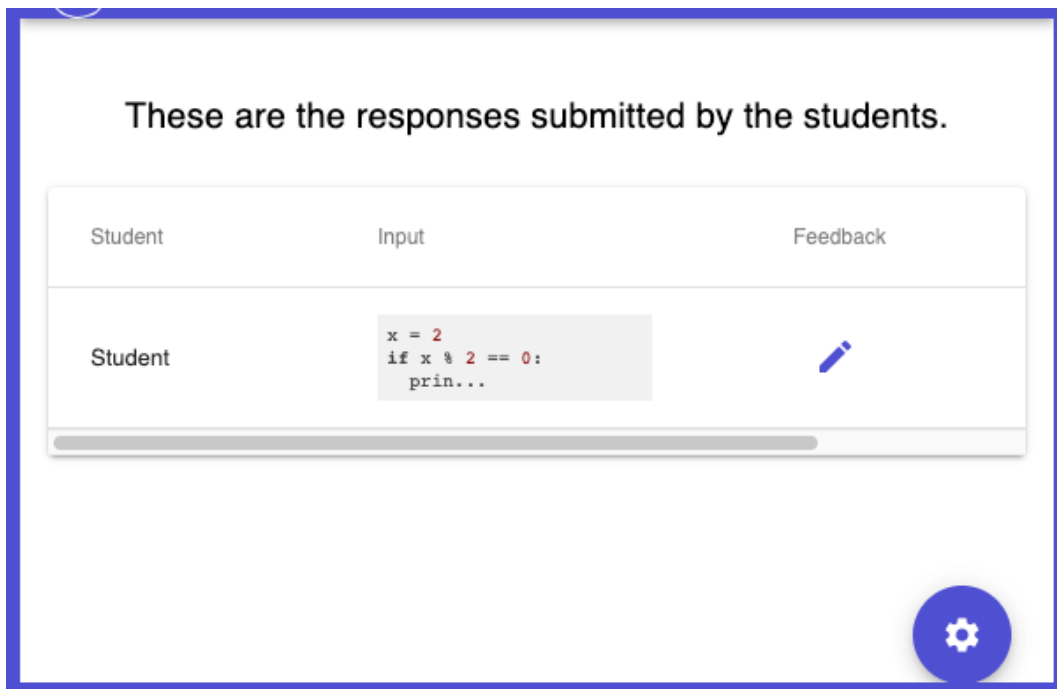


Figure 10. Contributions of all students as seen by the teacher in the authoring view.

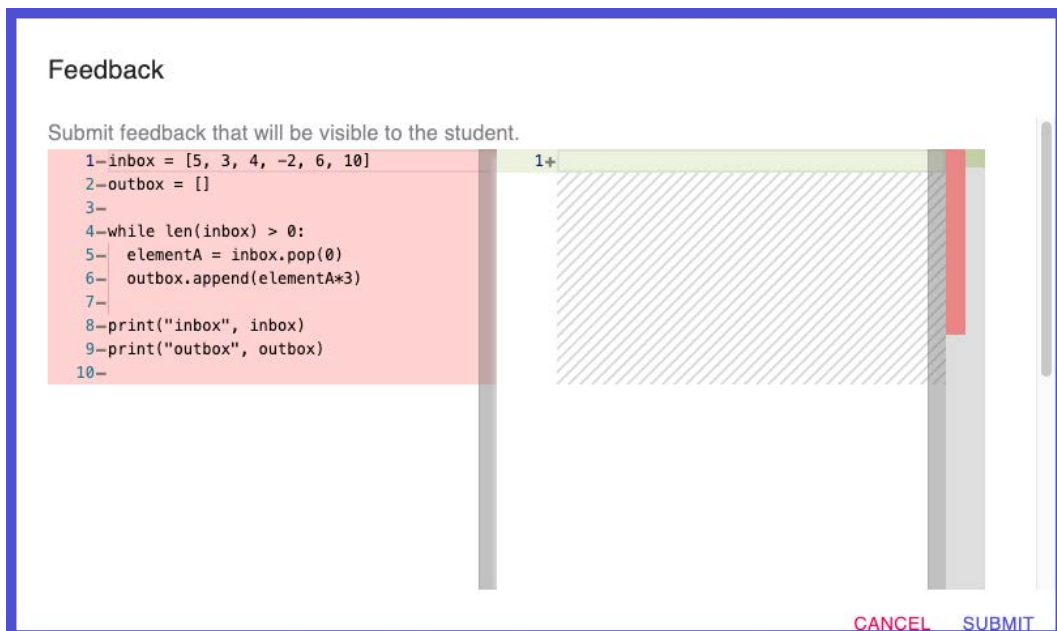
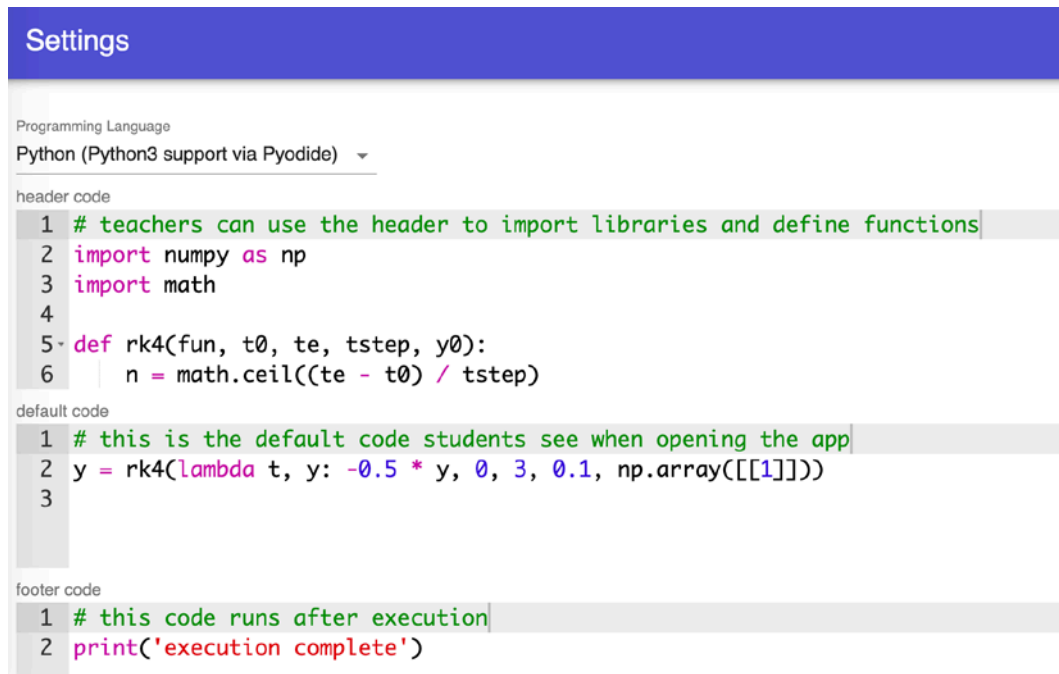


Figure 11. Contributions of a selected student as seen by the teacher in the authoring view and in the feedback mode (comments can be added in the hatched area).

In the next release of the Code app, it will be possible to load data files provided by the teachers to run data processing or machine learning algorithms (or more simple data visualization functions). In addition of printing text in the console, graphical outputs will be supported.

The example of the evolution of a population introduced in Section 4 and implemented using the Code app is presented in Fig. 12, where the Math and NumPy libraries are loaded as part of the header code. NumPy is adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.



The screenshot shows the 'Settings' window of the Code app. At the top, there is a blue header with the word 'Settings' in white. Below this, the 'Programming Language' is set to 'Python (Python3 support via Pyodide)'. The interface is divided into three sections: 'header code', 'default code', and 'footer code'. Each section contains Python code for a population evolution simulation.

```

Programming Language
Python (Python3 support via Pyodide)

header code
1 # teachers can use the header to import libraries and define functions
2 import numpy as np
3 import math
4
5 def rk4(fun, t0, te, tstep, y0):
6     n = math.ceil((te - t0) / tstep)

default code
1 # this is the default code students see when opening the app
2 y = rk4(lambda t, y: -0.5 * y, 0, 3, 0.1, np.array([[1]]))
3

footer code
1 # this code runs after execution
2 print('execution complete')

```

Figure 12. Settings of the Code app for the study of the evolution of a population.

The numerical Runge-Kutta integration algorithm (order 4) is also defined in the header code. The full Runge-Kutta code (hidden in Fig. 12) is:

```

def rk4(fun, t0, te, tstep, y0):
    n = math.ceil((te - t0) / tstep)
    y = np.zeros((n + 1, len(y0)))
    y[0, :] = y0
    for i in range(n):
        f1 = fun(t0, y0)
        f2 = fun(t0 + tstep / 2, y0 + tstep / 2 * f1)
        f3 = fun(t0 + tstep / 2, y0 + tstep / 2 * f2)
        f4 = fun(t0 + tstep, y0 + tstep * f3)
        t0 += tstep
        y0 = y0 + tstep * (f1 + 2 * (f2 + f3) + f4)
        y[i + 1, :] = y0
    return y

```

The code written by a student and the corresponding output is given in Fig. 13. The student has to define simply the parameters of the model and call the simulation function with the model itself and details about the time interval for the simulation.

```
1 # the student writes code in a separate editor
2 # but can use functions predefined by the
3 # teacher, such as `rk4` in this case
4
5 r = 0.2
6 K = 100
7 P0 = np.array([[1]])
8 P = rk4(lambda t, P: r * P * (1 - P / K), 0, 60, 0.2, P0)
9
10 print(P[0:10])
```

```
[[1.          ]
 [1.2423171 ]
 [1.54258527]
 [1.91424916]
 [2.3736495 ]
 [2.94052701]
 [3.63854773]
 [4.49580874]
 [5.54525507]
 [6.82490053]]
execution complete
$
```

Figure 13. Code written by a student to study the evolution of a population using a numerical integration function pre-defined by the teacher.

The GoModel and Code apps (amongst others) are essential to develop and strengthen the scientific understanding of the students. First, they help them to understand that all the online labs they are using from Golabz are based on models. Second, they can understand that simplified or more complex models of a same reality can be built and that they are just intellectual approximative representations. Third, they can understand that models require parameters to well (or not) represent the reality. Fourth, they can understand that **animation** of physical systems is based on **simulation**. Last but not least, if proper examples are selected, they can see that nature is usually nonlinear (like the evolution of a population) and that doubling a cause (action) will not always trigger a double effect (reaction).

6. Sysquake App

Sysquake is a standalone and mobile app developed by Calerga that emulates and complements the professional Matlab software package. According to Wikipedia, Matlab is “a multi-paradigm numerical computing environment and proprietary programming language developed by MathWorks. It allows matrix manipulations, plotting of functions and data, implementation of algorithms, ...”.

Sysquake pioneered the concept of interactive graphics which can be used to adapt models and parameters in real-time by moving directly with the mouse data points of time or frequency responses and getting the corresponding, reverse-engineered, parameters. This feature is very useful in education and research to understand the links existing between models and behaviours.

A version of Sysquake, fully implemented in JavaScript and running completely in the Web browser, was made available as a free app to be integrated in the Go-Lab ecosystem at the beginning of the project. While it targets students in higher educational levels than the GoModel app, it offers interesting complementary opportunities to bootstrap Computational Thinking activities.

One should also mention that Sysquake scripts can be turned into single-file HTML simulations which can be shared and run independently as virtual labs. This function is however not yet promoted and exploited in the Go-Lab initiative as it requires to master Matlab and Sysquake.

The population model introduced in Section 4 can be studied using Sysquake by typing the following commands:

- `r = 0.2;` (Assigning the value 0.2 to the parameter r)
- `K = 100;` (Assigning the value 100 to the parameter K)
- `Po = 1;` (Assigning the value 1 as the initial population)
- `(t,P) = ode45(@(t,P) r*P*(1-P/K), [0,60], Po);` (See **)
- `plot(t',P')` (Plotting the resulting population with respect to time t)
- ** Integrating the differential equation defined between the square brackets, within a time interval from 0 to 60, with the initial condition Po . The semicolon at the end of each command is to avoid displaying the result on the light-blue console (Fig. 14).

The corresponding sequence of commands introduced in the Sysquake app by the student is shown in Fig. 14.



Figure 14. Commands for the simulation of the evolution of a population with Sysquake.

When clicking on the Figure button, the graph generated by the plot command is displayed (Fig. 15).

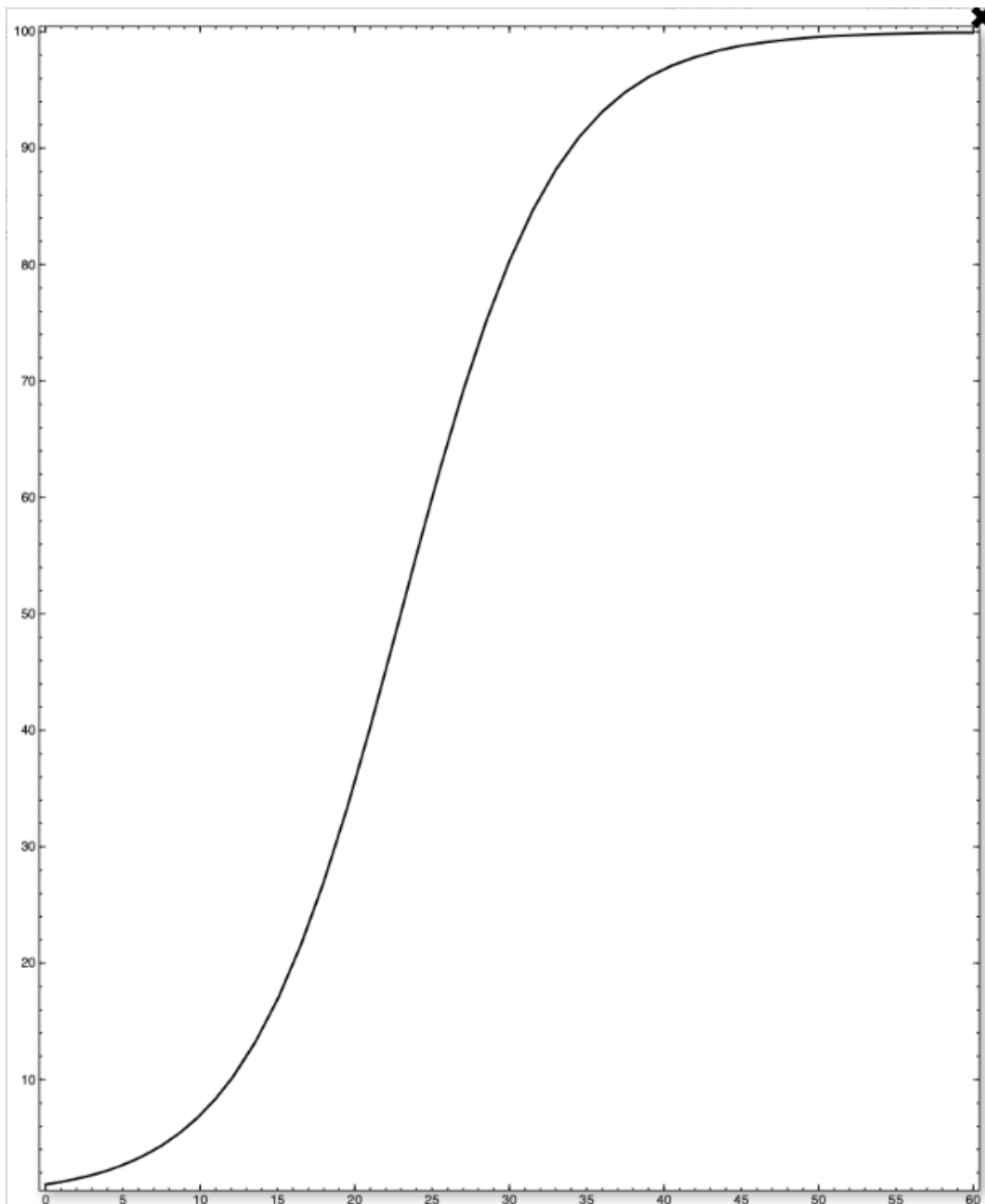


Figure 15. Results of the simulation of the evolution of a population with the parameters set in Sysquake by the student.

Section 4, 5, and 6 show complementary ways to study dynamical systems considering not only the parameters, but also nonlinear functions and models. The same type of studies can also be carried out in Python with the Code app providing the necessary numerical simulation libraries are used (Section 5).

7. GeoGebra App

GeoGebra is an interactive geometry, algebra, statistics and calculus application, intended for learning and teaching mathematics and science from primary school to university level. GeoGebra is available on multiple platforms with its desktop applications for Windows, macOS and Linux, with its tablet apps for Android, iPad and Windows, and with its web application based on HTML5 technology. You can find more information about GeoGebra on <https://www.geogebra.org/>.

Until now GeoGebra material could only be used by placing a link to it in the ILS, such as <https://www.geogebra.org/m/BbJsUCAV> (Add Link). An alternative is to create a dedicated lab, in which the GeoGebra material runs (Add Lab). The GeoGebra app (Add App) makes it easy to run a GeoGebra material in a frame with a blue header inside an ILS, by just specifying the desired GeoGebra material.

The teacher can find the desired GeoGebra material on the GeoGebra web site (<https://www.geogebra.org/materials>) and paste its resource URL in the configuration (see Fig. 16). After the resource URL has been pasted, the GeoGebra material is immediately loaded (see Fig. 17). The student view of the GeoGebra material is shown in Fig. 18.

GeoGebra material cannot be developed in the GeoGebra app. The GeoGebra app can only load existing GeoGebra material stored at the GeoGebra web site.

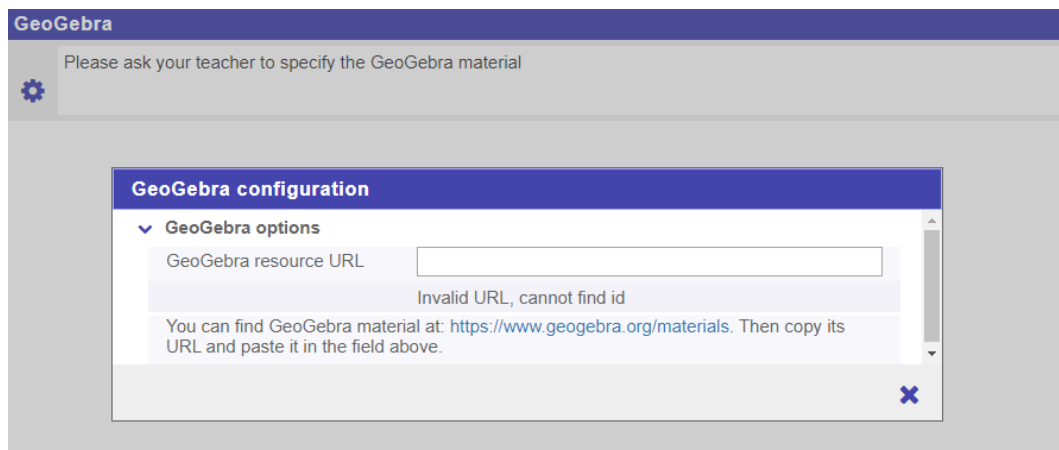


Figure 16. The empty GeoGebra configuration screen, with the student preview.

We have tried to store the current student state of the GeoGebra material. Unfortunately, retrieving the GeoGebra state takes some time. As a result, an automatic save action when the browser tab is closed will fail, because the tab is closed before GeoGebra returns its state. This means that a reliable auto save is not possible. A manual save action by pressing a save button is possible. However, as all other apps are auto saving, we decided not to implement a manual save option.

In a future version, the logging of student actions in the GeoGebra material could be implemented.

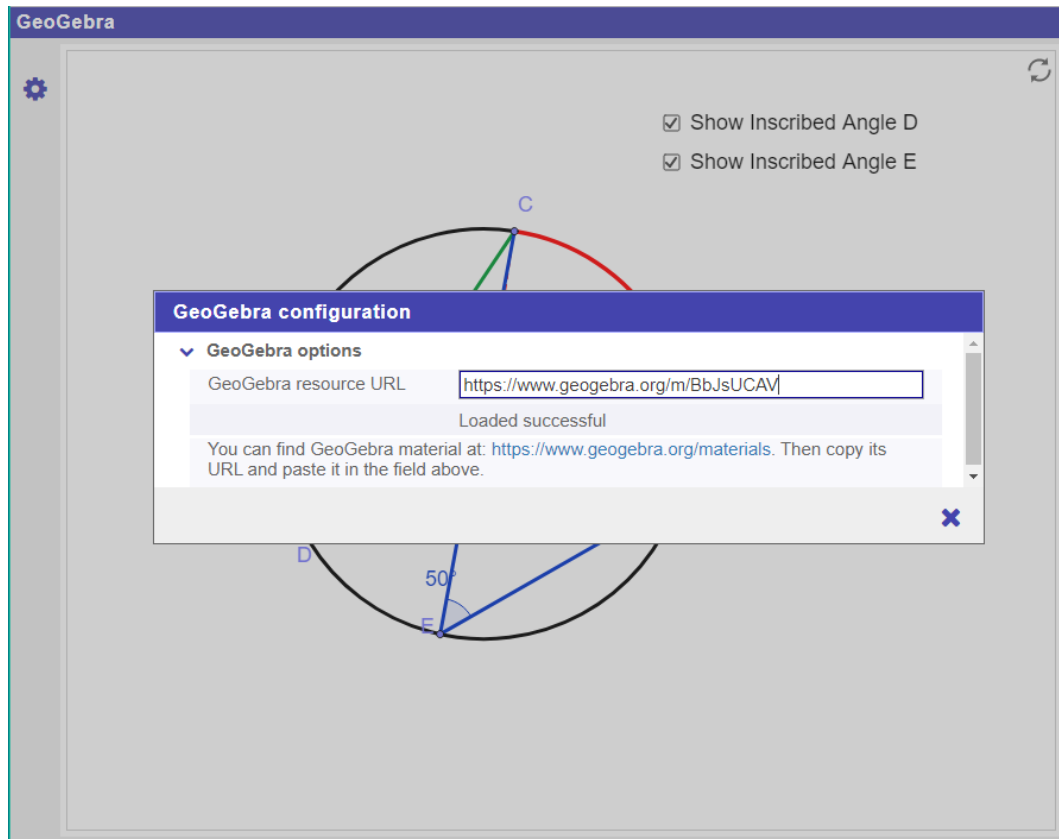


Figure 17. The filled in GeoGebra configuration screen, with the student preview.

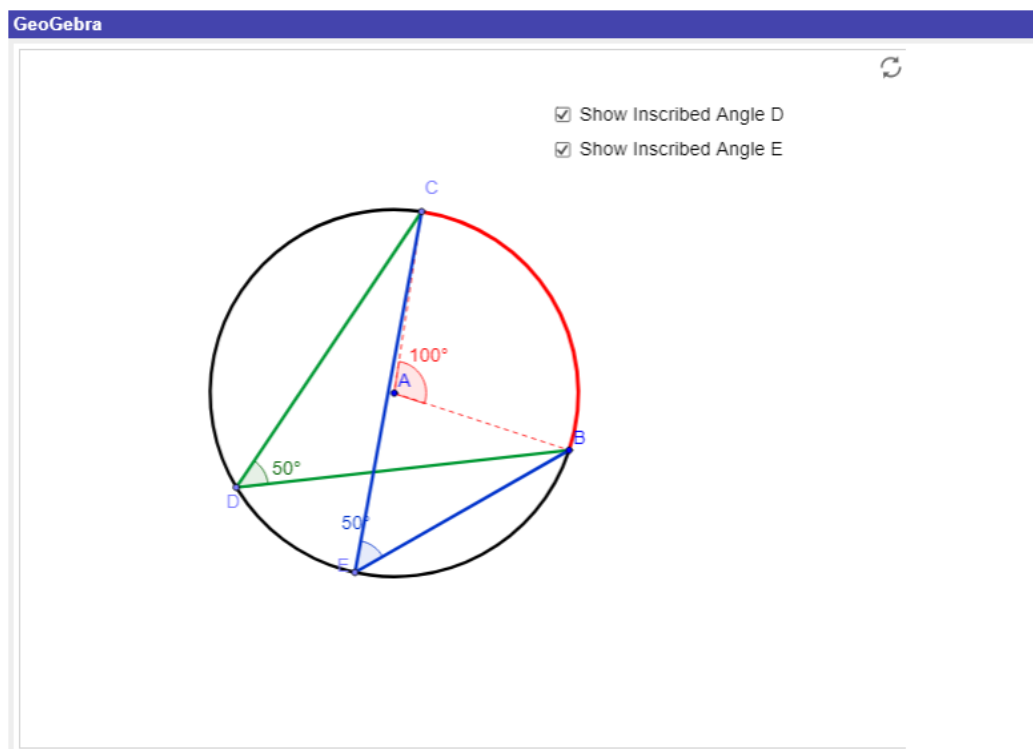


Figure 18. The student interactive view of the GeoGebra material.

8. Scratch App

Scratch is a block-based visual programming language and online community targeted primarily at children. Users of the site can create online projects using a block-like interface. The service is developed by the MIT Media Lab.

Until now, Scratch programs could only be used by placing a link to it in the ILS, but then the Scratch programs has to be used outside the ILS (except if the /embed extension is added to its link), such as <https://scratch.mit.edu/projects/123456/embed/> (Add Link). An alternative is to create a dedicated lab, in which the Scratch program runs (Add Lab). The Scratch app (Add App) makes it easy to run a Scratch program inside an ILS, by just specifying the desired Scratch program.

The teacher can find the desired Scratch program on the Scratch web site (<https://scratch.mit.edu/explore/projects/all>) and paste its resource URL in the configuration (see Fig. 19).

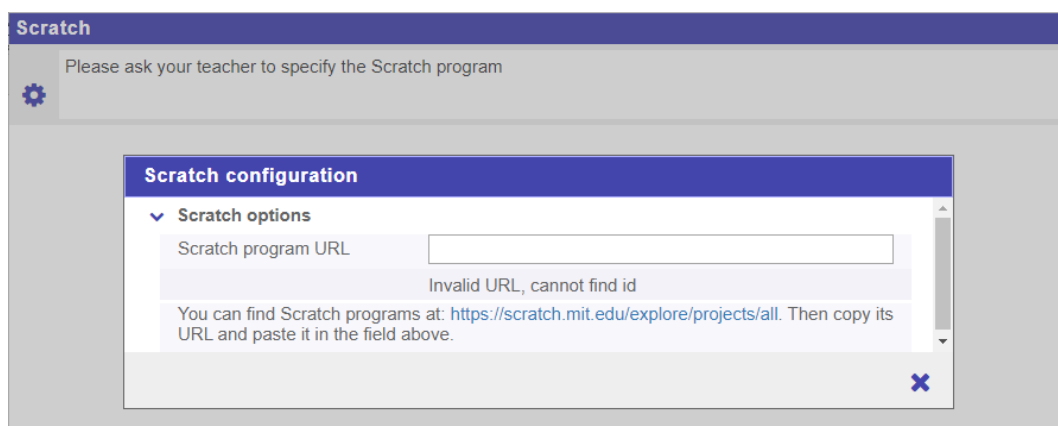


Figure 19. The empty Scratch configuration screen, with the student preview.

After the resource URL has been pasted, the Scratch program is immediately loaded (see Fig. 20). The student view of the Scratch program is shown in Fig. 21.

Scratch programs cannot yet be developed in the Scratch app. The Scratch app can only run existing Scratch programs stored at the Scratch web site.

Blockly from Google (<https://developers.google.com/blockly>) is another environment supporting graphical programming for children. MIT and Google are working together on the development of a new generation of graphical programming blocks, called Scratch Blocks.

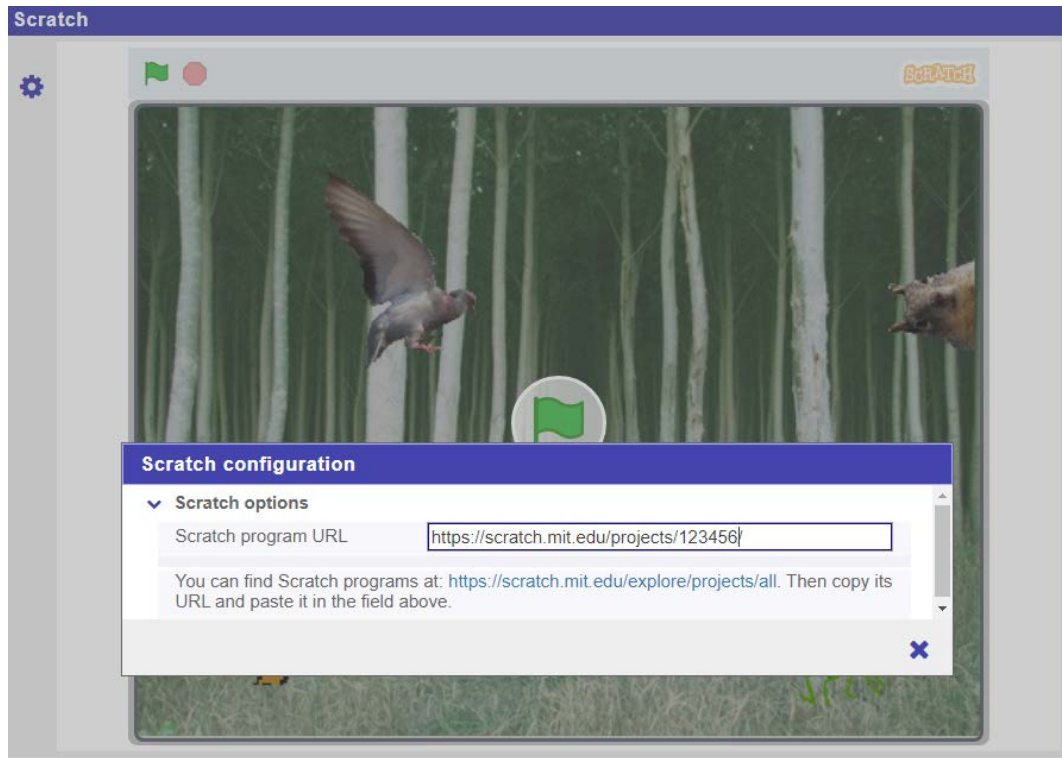


Figure 20. The filled in Scratch configuration screen, with the student preview.

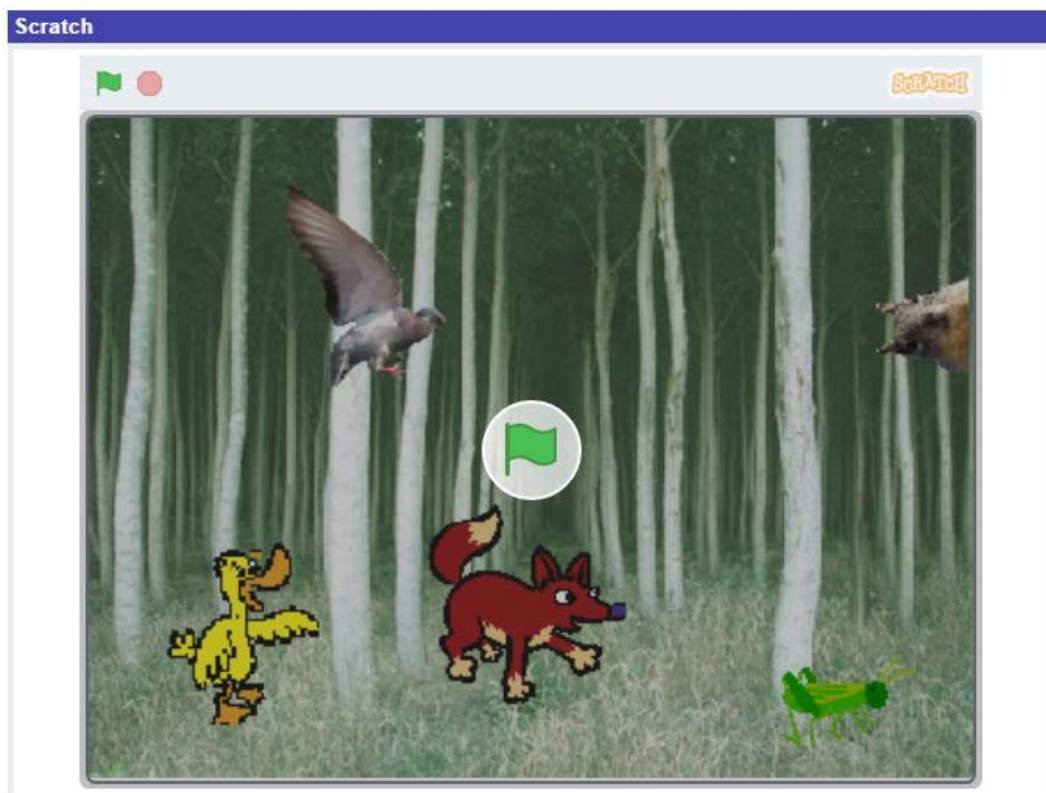


Figure 21. The student view of the Scratch program.

9. Conclusions and Outlook

The concept and the implementation of the ePortfolio features have significantly evolved from the initial definition in the Grant Agreement as a separated app to their current implementation as eBooks. This new direction is better aligned with the privacy-by-design approach implemented since the initial release of the Go-Lab ecosystem (with students only registered with nicknames) and with the new constraints regarding data protection guaranteed by the GDPR, especially the fact that personal data cannot be moved between platforms and domains without explicit consent (confinement).

New developments made to support off-line use of ILSs in the GO-GA project have paved the way for an even more exciting implementation combining a viewer and full-fledged ILS archives. The adoption of ePortfolio solutions by students has always been a challenge. The possibility offered to just get a document like a PDF file, an eBook, or an ILS archive executable in an open source desktop viewer, free students from the constraints to use a dedicated cloud platform for which they need perpetual support and credentials and that is not part of their personal digital ecosystem.

Regarding GoModel, as it is, we do not expect any changes in the future, except for the possible maintenance of the underlying external packages being used (i.e., mathematics formatting and graph drawing).

The study of dynamical systems using tools like GoModel is just the first step towards fully supporting Computational Thinking in secondary schools and beyond. The Code app recently proposed and the Sysquake app that was available since the beginning of the project can support the acquisition and the consolidation of computational thinking skills and hence broaden the dissemination and implementation scope of the Go-Lab initiative.

Links with other open initiatives for supporting digital math education (GeoGebra) and computational thinking (Scratch) are also integrated in the Go-Lab ecosystem as simple apps.