**Επιστημονική Επετηρίδα Παιδαγωγικού Τμήματος Νηπιαγωγών Πανεπιστημίου Ιωαννίνων**

Τόμ. 9, Αρ. 1 (2016)


# Learning Renewable Energy by Scratch Programming

*Ioannis Balouktsis, Gerasimos Kekeris*

doi: 10.12681/jret.8916



**Βιβλιογραφική αναφορά:**

Balouktsis, I., & Kekeris, G. (2016). Learning Renewable Energy by Scratch Programming. *Επιστημονική Επετηρίδα Παιδαγωγικού Τμήματος Νηπιαγωγών Πανεπιστημίου Ιωαννίνων*, *9*(1), 129–141. https://doi.org/10.12681/jret.8916

# Learning Renewable Energy by Scratch Programming

**Ioannis Balouktsis, Gerasimos Kekkeris**

Primary Education Department, Democritus University of Thrace

**Abstract**

Scratch is a simple, media-rich programming language that has been developed to support self-directed learning through exploration, tinkering and collaboration with peers. It is being used more and more from teachers and students as a tool for building scientific models and evaluating students' behavior in schools. Students, while sharing interactive projects, develop skills in areas such as acquisition and development of concepts, problem solving abilities, creative thinking, working collaboratively and all of that in a playful spirit. In this paper, we present Scratch as a useful tool for teaching renewable energy issues, mainly between the ages of 11 and 16, in order students to develop an effective understanding through interactive sustainability projects and to cultivate awareness and attitudes towards energy sustainability with new media. We also present how students can work on renewable energy and its types and control the experiments about solar and wind energy using Scratch. The key design goal of the project is to deep students' understanding in the world of renewable energy and to help students in interacting with solar and wind energy experiments.

**Key-words:** Scratch Programming Language, Renewable Energy, Solar Energy, Wind Energy

Correspondent Author: Dr. Ioannis Balouktsis
Department:, Primary Education Department, University: University of Thrace
e-mail: balouktsis1982@yahoo.co.uk
e-publisher**:** National Documentation Centre, National Hellenic Research Foundation
URL: http://ejournals.epublishing.ekt.gr/index.php/jret/index

**Introduction**

Over the past decade at MIT's Media Lab, Computer Science education researchers have developed Scratch, a notable project for teaching computer programming (Utting *et all* 2010). This programming environment emphasizes on interaction with media and focuses on teaching introductory programming concepts. Students learn to design, create, interact, and invent with new media (Resnick 2007) by using "blocks" that represent instructions, loops, decisions, variables, operators. So, through programming, students become able to develop computational thinking that helps them learn problem-solving and design strategies that carry over to non-programming domains (Wing 2006).

In contrast to programming languages that use complex syntax rules and seem to be hard and create barriers for novice programmers (Meerbaum-Salant *et all* 2010, Rizvi *et all* 2011, Woltz *et all* 2008, Woltz *et all* 2009), scratch allows students to interact with computation, tries to free them from syntax issues and offers to them the chance to focus on the programming logic (Woltz *et all* 2008, Malan and Leitner 2007, Maloney *et all* 2004, Resnick *et all* 2009). It uses a graphical programming environment that offers easy possibilities to with graphics, animation and sound. In scratch environment, students assemble graphical logic blocks to easily create media-rich programs full of their interaction, audiovisual digital media and no need to write source code manually (Woltz *et all* 2009, Maloney *et all* 2008). Students can quickly create interactive scenarios using scratch without much computer programming experience and with no prior computer programming knowledge. They need to focus on the structure of a problem and the logic program strategies to solve it and not be forced to deal with technical programming details that are not related to these main tasks. "Tackling all of these challenges simultaneously can be overwhelming and discouraging for beginning programmers" (Kelleher and Pausch 2005).

This approach, by combining blocks representing relational operations and including selection, inner/outer join, grouping, projection, aggregation and sorting, has a number of pedagogical advantages, such as simplification of the programming environment, encouragement of self-directed learning, visual execution of programs, elimination of syntax error messages, and more concrete data (Maloney *et all* 2010). Eleven categories of learning activities which can be performed within scratch, taking

into account social and constructivist learning approaches, are formulated. Computing teachers can use these categories of activities in their attempts to design appropriate everyday classroom settings for the learning of programming by novices within scratch (Koraki 2012). There are several investigations of the usability of scratch in specific scenarios of various education fields and in creating interactive stories into projects by students with no prior experience in scratch (Quan 2013, López and Hernández 2015, Balouktsis and Kekkeris 2014). This paper looks at usability aspects of scratch in the acquisition and development of renewable energy concepts in 11 to 16 year-old students: The creation of interactive scenarios about renewable energy types, solar and wind energy, for students with no prior experience in scratch.

**Scratch programming language as a didactic tool for teaching renewables**

Scratch is a visual programming language and is being used increasingly more at schools (Monroy-Hernández and Resnick 2008, Maloney *et all* 2010). Students can program with a mouse by assembling blocks as language elements to build programs, that only fit if syntactically appropriate. The building blocks follow a color code and this makes a number of key concepts more visible and understandable and also it makes easy for students to learn about relational operations by using a hands-on manner. Students can also easily import graphics and sounds and using these elements they can create interactive multimedia projects and visualize and analyze interesting data sets. Animation and sound can be controlled by code. Code can be used also to react to input from external sensors that can be attached via USB (Malan and Leitner 2007).

The core design principles for scratch are it to be made more tinkerable, more meaningful and more social than other programming environments (Resnick *et all* 2009). With scratch sensor board, students can create interactive projects that allow them to sense and react to events in the physical world. Therefore, scratch is suitable for encouraging students' exploration and generation of explanations by themselves, as they can express their own ideas, specifying which variables intervene in a phenomenon and investigating how these variables could be related López and Hernández (2015).

Probably the biggest challenges for scratch are not technological but cultural and educational (Margolis 2008). Scratch has been a success among early adopters. Its use expands the notion that digital fluency includes not only browsing and interacting but also designing and creating. More broadly, scratch helps to be a shift in how people think about programming in general.

In a classroom setting, teachers can use scratch for a variety of purposes ranging from interactive demonstrations to homework assignments. For example, students can discuss in class with classmates and/or teachers and then develop scientific conceptual models by approaching and developing scientific phenomena and focusing on the nature of the relationships between the components of the model.

To conclude, we consider that scratch offers many opportunities as an activity to teach and learn renewables. The educational potential of this tools is important and teachers should include it in their teaching practice. So, in our case of renewables, we propose the use of scratch for this didactic purpose but also its general use in education.

## Interactive scenarios

Nowadays, renewable energy is coming to be more and more important issue, due to the fact that energy sources and producing energy technology are in the forefront and they are demanding persons skilled in this field. Not surprisingly, a compulsory class can be a jarring disappointment to new students expecting to play and understand issues in this field, because renewables' software programs developed for educational use are limited, dull and antiquated. To address these issues, we have created fully interactive scenarios, where students can interact and study via Scratch environment with renewable energy sources and have an impact of using visual data on their learning.

### *A simulation model on solar energy*

Through the annual education process Greek students of the secondary school must be taught about the importance and use of the various types of renewable energy sources and must be able to identify affordable and practical solutions on renewables. The

first scenario shows how it is possible, by using the scratch software and developing simulation applications, to develop a relatively simple application project in the field of solar energy systems.

Figure 1 displays the project's visualization window, which consists of three sprites (sun, clouds and a pv panel). When the green flag is activated, the variable monitor of pv power calculates the power in W / $m^2$ of the pv panel. In calculating the force of pv panel, it is considered the assumption that the efficiency remains constant and equal to 15% and that solar radiation in a plane perpendicular to the sun's rays with fine sky, is equal to 1000 W / $m^2$. The user is able to adjust a) the inclination of the pv panel relative to the horizontal plane, (pv tilt slide monitor), b) the angle of the sun's rays relative to the horizontal plane (sun angle slide monitor) and c) the percentage of cloud cover at the sky (cloud cover slide monitor).

It is therefore a model that describes the output power per square meter of a P / V panel by utilizing the direct incident solar radiation to it, allowing the investigation possibility of how it is affected from the tilt angle of the P / V panel, the angle of sun height and the clearness of the sky. How can a model be built with scratch to describe a solar system as it is described above?


**Figure 1.** A screenshot that shows the backdrop and the three (3) sprites (sun, clouds and solar panel). The model can be accessed online at https://scratch.mit.edu/projects/94433100/

First of all, it is required the mental models to be expressed and the variable models, relationships and conditions implied to be explained, in order students to be able to identify the variables that are involved in the model (in this case, sun position, angle of sun height, angle of pv panel tilt, per cent of cloud cover and pv power output) and assign initial values (the initial conditions) to these variables. Afterwards, it must be established the set of relationships between variables.

To build the above variables and relationships into a computational model by using scratch language, it is necessary to build programming blocks like those in figure 2.

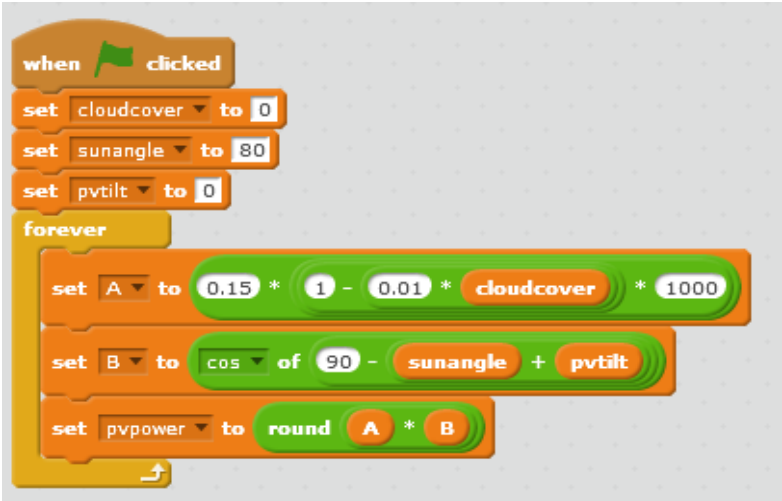



**Figure 2.** Programming blocks determining the output power of pv panel and defining the initial conditions of variables.

Secondary school students should be able to build with the proper support after becoming familiar with the program's language. First of all, they should define the initial conditions for the variables each time the program restarts by activating on the small green flag in the top right hand corner of figure 2.

Also, at the programming blocks in figure 2, it is shown the calculation of the output power of the PV panel as a result of the direct solar radiation incidence on the panel. In this case it is chosen the simple way of calculation and not the complex one, in which we should take into account the incidence of diffuse radiation and reflection of incident solar radiation in the plane of photovoltaic. This method is applied for two reasons. Firstly, the programming should be easier for students and secondly to emphasize the qualitative interpretation of the dependence of output power on the angle of incidence of solar radiation at the level of a photovoltaic panel and clearness

of the atmosphere. In the simple case, where a photovoltaic has southerly direction with the sun in the solar noon, the power output of a solar panel, taking into account only the direct component of solar radiation, is given by:

$$P = n * I * \cos(a) = n * I * \cos(90 - (s + b)) \quad (1)$$

wherein ***n*** is the efficiency of the photovoltaic, ***I*** is the direct component of the solar radiation in a plane perpendicular to PV panel, ***a*** is the angle of incidence of solar radiation in the plane of the photovoltaic, ***s*** is the tilt angle of the PV panel relative to the horizontal plane and ***b*** is the angle of sun rays relative to the horizontal plane (see figure 3).

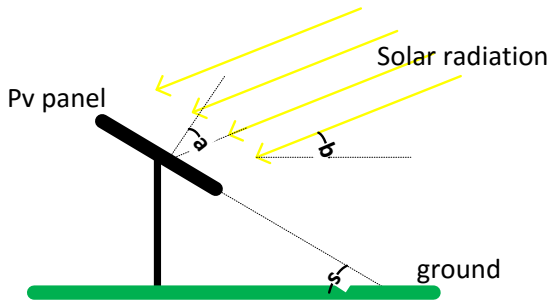
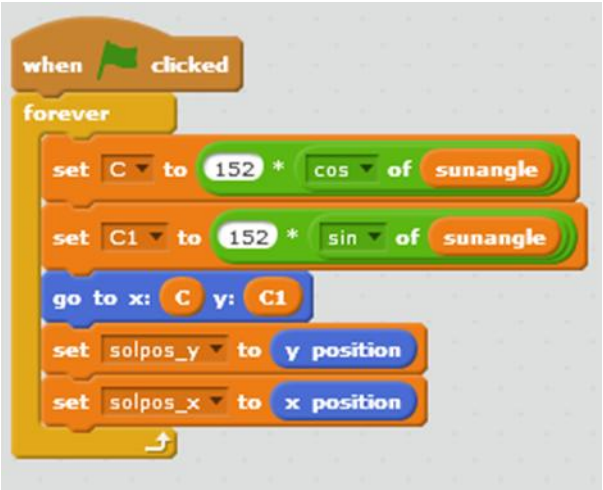



**Figure 3.** The angles that are referred at eq. (1) during the incidence of solar radiation to a PV panel

In order the user to has the ability to change a) the slope of the pv panel, b) the position of the sun (hence the angle of the sun's rays) and c) the coverage rate of sky with clouds (cloud cover slide monitor), three sprites were created (sun, cloud and pv module). Figure 4 shows the programming blocks for sprites sun and cloud.

In figure 4 it is also shown the code of the program that moves the sprites of sun and cloud according to the values given by the user for the sun's angle (sunangle variable) and the inclination of photovoltaic (pvtilt variable). The user changes the values of variables as shown in figure 1 by means of sliders. Also the user can alter the clearness of the sky through cloudcover variable whose value changes using a slider as shown in figure 1. The cloud cover of the sky is implemented through the cloud sprite which utilizes traffic characteristics, change shape features and different

transparency levels. These functions are achieved using the clone blocks.



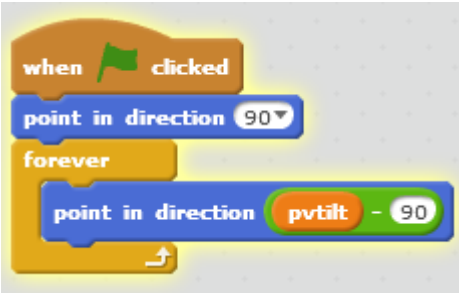




**Figure 4.** Programming blocks that configure the motion of the sun and PV panel according to the user's input

Cloning as a feature allows the creation of a clone by a sprite, of itself, or a semi-duplicate one, while the project is running. Clones of a sprite are the same as the original parent sprite, but they are a separate instance and even if they inherit the parent's scripts and all parent's properties, they can then be modified.

The code for the sprite cloud is not shown in the paper because it is kind of complex and can be found by visiting the link https://scratch.mit.edu/projects/94433100/ When users have defined all variables with the values they wish, then the output pv power shows the resulted photovoltaic power in W / $m^2$ as shown in figure 1 and which is calculated by using the formula 1. From here, one can rebuild a more complex model by adding motion between east and west to the sun sprite, not only during solar noon. Also more complex calculation formulas could be used for estimating solar radiation at the level of pv panel and the corresponding power that is produced. Finally, the possibility of downloading solar radiation data from online sites and use them directly in the simulation model of scatch could be researched.

*A simulation model on wind energy*

The second scenario shows how it is possible, by using the scratch software and developing simulation applications, to develop a relatively simple application project in the field of wind energy systems.

Figure 5 displays what appears in the project's visualization window, which is constituted by three sprites (in this case, the 3-blade wind turbine, the 6-blade wind turbine and the god of wind Aeolus). When the green flag is activated, the variable monitor of w_gen power shows the power in W of a wind generator. The user is able to set a) the wind speed b) the turbine efficiency and c) the size of the blades of the wind turbines. During the simulation, two identical turbines are used, but with different blade number so that the user can see the dependence of the rotation frequency by the blade number. It is therefore a model that describes the power output of two identical wind turbines in W by utilizing wind kinetic energy, enabling the user to explore how the power output is affected from the wind speed, the size of the blade and the coefficient output of the wind turbine.

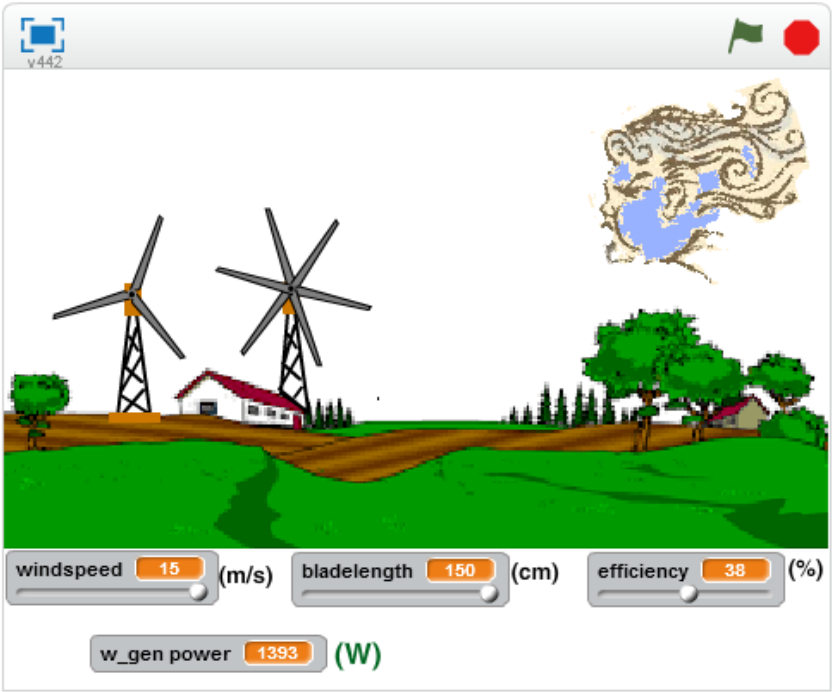



**Figure 5.** A screenshot that shows the backdrop and four (4) variables of the simulation model. The model can be accessed online at https://scratch.mit.edu/projects/94433466/

The equation that calculates the force is given by:

$$\boldsymbol{P = c_p \frac{1}{2} \rho \pi R^2 v^3} \qquad (2)$$

wherein $\mathbf{c_p}$ is the efficiency degree of the wind turbine, $\boldsymbol{\rho}$ is the density of the air, $\pi \boldsymbol{R}^2$ is the surface of the blade and **u** is the wind speed. In order the user to have the ability to change a) the wind speed, b) the size of the blade, and c) to observe the blades rotation of the wind turbines with different rotational frequency due to the different number of blades, three sprites were created (3_blades, 6_blades and Aeolus). In figure 6 the programming blocks to sprites 3_blades and Aeolus are depicted.


**Figure 6.** Programming Blocks that rotate the blades of the 3-blade turbine and produce the sound of the wind blow by simultaneously varying the size of the Wind God's head

The rotation of the blades of both turbines is considered proportional to the wind speed and inversely proportional to the number of blades and size (Twidell and Weir 2006).

$$\boldsymbol{turn\ of\ blades} \sim \frac{\boldsymbol{v}}{\boldsymbol{n\,R}}$$

where $\boldsymbol{n}$ the blade number.

The user can toggle the values of variables via sliders as shown in Figure 5. From here, one can make the model more complex by adding the option of setting manually the wind direction and employing more complex calculations of wind turbine's efficiency degree. Finally, it could be researched the option of downloading speed data from Internet and utilize them directly in the simulation model of scratch.

## Conclusions

The two scenarios on renewable energy sources include innovative learning activities within the context of scratch environment. Secondary school students can use such scenarios utilizing scratch programming language to modify and rebuild their own models, by specifying and adding variables involved in a phenomenon and, researching relations between them. Moreover, student's learning on energy sustainability is encouraged congruously with the development of their intellectual skills in a playful and creative way through collaborative learning activities. So, students are able to make predictions and evaluate and discuss the results of their running project with classmates and/or teachers in a scientific way.

Teachers can use scratch scenarios and take full advantage of a digital innovation like this to design appropriate every day classroom scenarios. This expands their classes not only by creating environments for the students to build and evaluate renewable energy models but also encouraging them to apprehend energy sustainability parameters and research their relations between them.

**References**

Balouktsis, I., & Kekkeris, G. (2014). A Science Concept on Electricity Supported by the Scratch Programming Language. In *Proceedings of the YILDIZ International Conference on Educational Research and Social Sciences,* YICER- 2014, pages 114-121. Istanbul, Turkey.

Kelleher, C., & Pausch, R. (2005). Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. *ACM Computing Surveys,* 37, 83-137.

Koraki, M. (2012). Diverse categories of programming learning activities could be performed within Scratch. WCES-2012, *Procedia-Social and Behavioral Sciences,* 46,1162-1166.

López, V., & Hernández, M.I. (2015). Scratch as a computational modelling tool for teaching physics. *Physics Education,* 50 (3), 310-316.

Malan, D. J., & Leitner, H. H. (2007). Scratch for budding computer scientists. In *Proceedings of the 38th SIGCSE technical symposium on Computer science education*, pages 223-227. Covington, Kentucky, USA.ACM.

Maloney, J., Burd, L., & Kafai, Y. (2004). Scratch: A Sneak Preview. In *Proceedings of the Second International Conference on Creating Connections and Collaborating,* C5'04, pages 104-109. Washington, DC, USA.

Maloney, J., Peppler, K., Kafai, Y., Resnick, M., & Burd, L. (2008). Programming by choice: urban youth learning programming with scratch. *SIGCSE Bulletin*, 40, 367-371.

Maloney, J., Resnick, M., Rusk, N., Silverman, B., & Eastmond, E. (2010). The Scratch Programming Language and Environment. *Transactions on Computer Education,* 10(4), 16:1-16:15.

Margolis, J. (2008). *Stuck in the Shallow End: Education, Race, and Computing*. The MIT Press. Cambridge, MA.

Meerbaum-Salant, O., Armoni, M., & Ben-Ari, M. M. (2010). Learning computer science concepts with scratch. In *Proceedings of the Sixth international workshop on Computing education research,* ICER '10, pages 69-76, New York, NY, USA.ACM.

Monroy-Hernández, A., & Resnick, M. (2008). Empowering kids to create and share programmable media. *ACM interactions*, 15(2), 50-53.

Quan, C. G. (2013). Continuing with the promise of "scratch" in the applied linguistic classroom. 13th International Educational Technology Conference. *Procedia-Social and Behavioral Sciences,* 103,245-254.

Resnick, M. (2007). Sowing the Seeds for a More Creative Society. *Learning and Leading with Technologies*, 35(4), 18-22.

Resnick, M., Maloney, J., Monroy-Hernandez, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., & Kafai, Y. (2009). Scratch: programming for all. *Communications of the ACM*, 52 (11),60-67.

Rizvi, M., Humphries, T., Major, D., Jones, M., 7 Lauzun, H. (2011). A CS0 course using scratch. *Journal of Computing Sciences in Colleges*, 26(3),19-27.

Twidell, J., & Weir, T. (2006). *Renewable Energy Resources.* Abingdon: Taylor & Francis Group. Second Edition.

Utting, I., Cooper, S., Kolling, M., Maloney, J., & Resnick, M. (2010). Alice, Greenfoot, and Scratch – A Discussion. *Transactions on Computer Education*, 10(4),17:1–17:11.

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

Woltz, U., Leitner, H. H., Malan, D. J., & Maloney, J. (2009). Starting with scratch in CS 1. In *Proceedings of the 40th ACM technical symposium on Computer science education,* pages 2-3, Chattanooga, TN, USA.ACM.

Woltz, U., Maloney, J., and Pulimood, S. M. (2008). 'scratch' your way to introductory cs. In *Proceedings of the 39th SIGCSE technical symposium on Computer science education,* pages 298-299, Portland, OR, USA.ACM.