ISBN: 978-989-8533-81-4 © 2018

GAMESONOMY VS SCRATCH: TWO DIFFERENT WAYS TO INTRODUCE PROGRAMMING

Cristina Rebollo Santamaría, Carlos Marín Lora, Inmaculada Remolar Quintana and Miguel Chover Sellés
Institute of New Imaging Technologies, Universitat Jaume I, 12071 – Castellón, Spain

ABSTRACT

The first obstacles to overcome when a student has to face the task of programming for the first time are the abstraction level, the comprehension of a language with unfamiliar concepts for him/her and the specific syntax for each programming language. This work presents the qualitative results obtained in a study focused on the gain of skills for learning programming languages. The experiment has been carried out in a classroom with 50 university students from the second year of the Degree in Video Game Design and Development. Students have worked with Scratch programming language on their first year, a tool created to facilitate programming learning, and with Gamesonomy, a game engine created to ease video game development without programming knowledge. The main purpose of this work is focused on the question: which one of these two tools is more appealing, easy to understand and to use, as well as which one of them is more efficient for a further initiation on the traditional programming. To test them, the students have fulfilled an evaluation test for each tool. After the analysis of the results, it is concluded that Gamesonomy seems to be more efficient to develop logical thinking and to have a better transition to conventional programming languages.

KEYWORDS

Game Engine, Visual Programming, Logical Thinking

1. INTRODUCTION AND LITERATURE REVIEW

The ability to code computer programs is an important part in today's society. One of the main problems to face a student on his/her first year computer science, comes from the abstraction and the inherent complexity of a programming language with unknown concepts such as variables, loops, matrices, functions and rest of specific syntax for each different programming language.

In order to comprehend any programming language, it is essential to develop a logical thinking. In 2006 (Wing, J., 2006), the computational thinking concept was defined as "problem solving, system design and understating of human behavior, by using the fundamental concepts of computer science". According to Grover and Pea (Grover, S., Pea, R., 2013) programming is not only a fundamental ability of computer science and a key tool to develop cognitive tasks involved in computational thinking, but also a way to improve students' thinking skills.

We could consider two different programming ways: conventional programming (Van Roy P. et al., 2003), (Jackson M., 1980) and visual programming (Chang, S.K., 1990), (Good, J., 2011). The first one is based on writing code, and the second one is based on a friendly visual environment easy to use (Repenning, A., 2017). Some examples of these are Java, C, C++, C# and Python for the conventional ones and Scratch and App Inventor for the visual ones. More recently, the game engines (Gregory, J., 2014) came out as another kind of tools with visual environments who can contribute to learn to programming without coding. Some examples of these engines are GameSalad and Gamesonomy.

As the usage of a visual environment for programming facilitates the understanding of conventional programming concepts, the research of this study is focused on evaluating the effectiveness of the educational method of teaching with a visual programming language or a game engine as first step before learning conventional programming. For this study, Scratch (ScratchTool), (Resnick, M. et al, 2009) has been selected as visual programming language and Gamesonomy (GamesonomyTool), as game engine. The reasons for selecting them were, for one hand, Scratch was conceived to ease programming learning and nowadays it is one of the most extended visual programming languages due to its ease of use, and for the other hand, the ease of use and comprehension of Gamesonomy which was created to facilitate game development for non-programmers. This game engine deals with the learning process from a ludic point of view, in order to avoid the initial non-acceptance usually present on many conventional programing environments. It worth to be highlighted, that both tools help to reach a better understanding of computational concepts and ease the logic programing thinking.

However, which one is seen as more useful by the students? With the purpose to answer this question, an experiment has been carried out with 50 second year university students from the Degree in Video Game Design and Development. The students have been working with Scratch and with the Gamesonomy's Game Logic Editor on their first year. The subjects where these kinds of software have been used are semester subjects, they are not programming subjects, and students have no previous programming knowledge. For this purpose, the evaluation has been conducted by an acceptance test (Davis, F.D., 1989), (Davis, F.D., Venkatesh, V., 2004) with the same method used by Zarraonandia et al, (Zarraonandia T. et al., 2017).

The work presented on this paper is organized as it follows. On chapter 2 both tools will be described and compared, after that on chapter 3 a simple example will be performed to compare them, later on chapter 4 the methodology of the test and the results of the study on the students will be presented and discussed. Finally, on chapter 5 the conclusions and the future work will be presented.

2. DESCRIPTION OF THE TOOLS

The two analyzed tools, Scratch and the Gamesonomy's Game Logic editor, have several features in common: both are visual programming environments, are perceived as easy to use, have multiplatform support, improve the computational thinking, teach programming fundamentals and let to check the actions that the user is programming. Furthermore, the student learns mathematic concepts with them such as: space coordinates, variables, algorithms, randomness, etc. Besides that, the main differences between these tools came from the Gamesonomy architecture, in which there are no loop structures because the system is evaluating all the events in a continuous loop. Also, it has no dependency on complex data structure and has no need of logic expressions. Next, a presentation of both tools is detailed, emphasizing their features and functionality.

2.1 Scratch

Scratch is a programming language designed to ease the introduction to programming (ScratchTool). The system is made up on a visual programming environment where the user can learn about the coding syntax in an intuitive way. It uses a drag and drop technique with graphical blocks in order to create programs ready to make animations, interactive storytelling, games, interfaces and presentations.

Scratch objects or sprites are configured using scripts, having only their position and size as properties. Its functionality is based in the usage of a set of actions or behaviors to specify the performance of some graphical content or even some peripheral device. These actions and behaviors have a graphical puzzle shape, making the programing task very similar to fit the puzzle pieces together. This philosophy eliminates one of the main obstacles for the students when facing the coding for first time: the uncomfortable and unfamiliar aspect of the programing environments. The actions and behaviors are grouped into different categories of scripts, which in turn offer a drop-down list with all the different options that allow you to configure the specific script action or behavior to be performed. For each one of the blocks categories there is a color to ease its recognition. These categories of actions and behaviors offered by Scratch are defined below:

- Motion: To translate and rotate an object on the screen.
- Looks: To change the object visual aspect: image, size, etc.
- Sound: To play or stop audio sequences.
- Pen: To draw specific color, shadow or line thickness.
- Data: To create new variables and links it to the program.
- Events: Handlers to trigger specific events.
- Control: Conditionals such as if, else, and so on ... and loops such as forever, start and stop.

- Sensing: To manage peripheral sensors and their inputs.
- Operators: Mathematical, logics, random and position getters.
- More blocks: Own blocks and external controllers.

2.2 Gamesonomy

Gamesonomy is a game engine devised to facilitate the videogames creation and design (GamesonomyTool). It uses a very intuitive interface where no coding is required. In the same way as Scratch, it entails a visual programing environment aimed to teach coding in an intuitive mode. It is also used the drag and drop technique for the buttons representing actions and conditions, these buttons are arranged on a graphical decision tree where a specific behavior is defined.

Gamesonomy works under the concept of Actor. Every object present on scene is an Actor; it carries a set of properties and a list of rules. These rules are built from the combination of the actions and conditions in order to perform a specific behavior. Each one of these actions and conditions are represented by a button on the Game Logic Editor to ease its recognition. The rules are graphically represented by a flow chart created from the arrangement of the actions and conditions in it. To arrange an action or a condition, the user has to drag the button from the lateral menu to the flow chart, and then place it in the desired place. This way, it makes that the student is programing while is playing, making him/her to forget he/she is actually writing code. The actions on the Game Logic Editor define the behavior of the Actors. On Figure 1, the set of available actions is presented and they will be described in the following lines:

- Edit: To change every parameter by a specific value or expression. It works in the same way for game, scene or object properties.
- Animate: Animation setter and controller. It executes animations by adding and arranging sprites and setting the frames per second rate.
- Destroy: To delete the object from the scene when is triggered.
- Spawn: An automatic object copy generator.
- Play sound: To activate a sound in the game.
- Move: To translate the object a certain quantity of units on screen. It has a related action called Move To: To travel towards a specific position or object.
- Rotate: To rotate the object a certain number of degrees. There is also a Rotate to action.
- Push: To apply forces on the object. Also it has a related action called Push To and another one working with the same concept to apply angular forces called Torque.

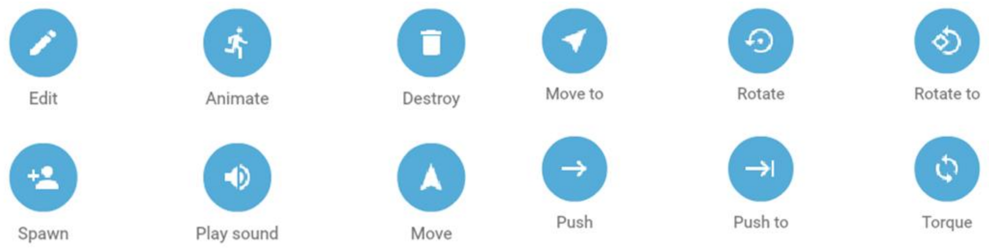


Figure 1. Actions used for the game logic

Conditions are the event triggers. They define the actions to perform on a decision making determined by an occurrence. It has been defined just six as it is shown in yellow also at Figure 2.

- Check: To check if a boolean property is met.
- Compare: To compare two data values from some game, scene or object features.
- Collision: To check if two objects are colliding. It relies on the object's colliding shape.
- Timer: To perform actions after a determined amount of seconds.
- Touch: To manage user interaction with mouse or touch events through tactile devices.
- Keyboard: To check which key has been pressed on the keyboard.

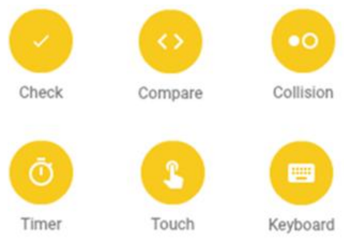


Figure 2. Conditions used for the game logic

Also, both conditions and actions are ready to work with numerical expressions and with mathematical functions: sin, cos, tan, asin, acos, atan, sqrt, random, and so on ... and the data types supported by this system are numbers and booleans.

3. PROGRAMMING EXAMPLE COMPARING BOTH TOOLS

Some code has been developed in order to understand the philosophy and the functionality of both tools. It has been tested with a mouse-following behavior: editable velocity conditions the time it takes an object to move from its original position to the point where the user has clicked the mouse. First of all, the pseudocode that solves the example that has been later programmed in Scratch and Gamesonomy is presented. This pseudocode is included only for the intended of performing a better understanding of the code. This section compares the way both programming environments face this example, but both tools allow the development of complete videogames.

3.1 Pseudocode

Figure 3 illustrates the necessary pseudocode to perform the example behavior. Let Player be the object to move across the screen and Mouse be the object that indicates the point in the screen where the object has to move to. Both have the position (x,y) property and the Player also has the dimension (size) and velocity properties by default.

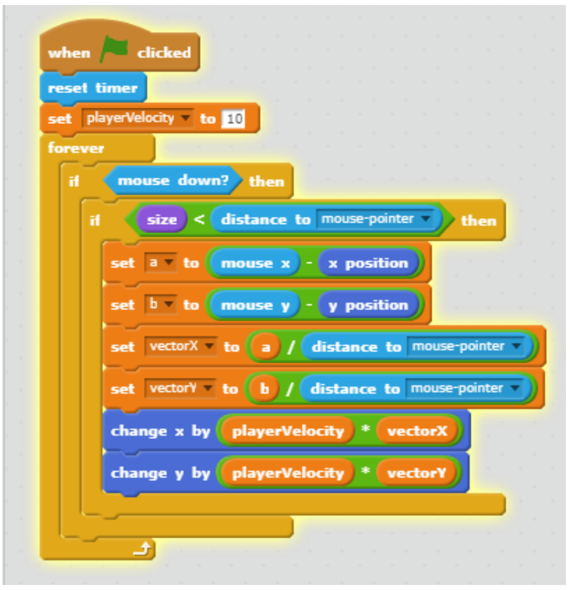
```
OnClick {
    var x = Mouse.x - Player.x;
    var y = Mouse.y - Player.y;
    var distance = sqrt(x*x + y*y);
    if(distance < Player.size) {
          var directionVector= { x: x / distance, y: y / distance };
          Player.x += Player.velocity * directionVector.x;
          Player.y += Player.velocity * directionVector.y;
      }
}
```

Figure 3. Pseudocode of the action performed as example.

Each time the environment registers a click event on the screen, the spatial coordinates are stored on Mouse.x and Mouse.y. Then, the unitary direction vector from the Player position to the Mouse position is determined by calculating the distance between these two points. If the required movement is greater than the dimensions of the Player, this action is performed. Then, the direction vector is obtained for each component and finally the movement is performed taking into account the Player velocity previously set.

3.2 Programming Language Scratch

The previous example code has been implemented with the programming language Scratch. For this purpose, it has been arranged a set of blocks to compose an instruction puzzle as it is shown in Figure 4.



```
when clicked
reset timer
set playerVelocity to 10
forever
    if mouse down? then
        if size < distance to mouse-pointer then
            set a to mouse x - x position
            set b to mouse y - y position
            set vectorX to a / distance to mouse-pointer
            set vectorY to b / distance to mouse-pointer
            change x by playerVelocity * vectorX
            change y by playerVelocity * vectorY
```

Figure 4. Action programmed with the Scratch Editor

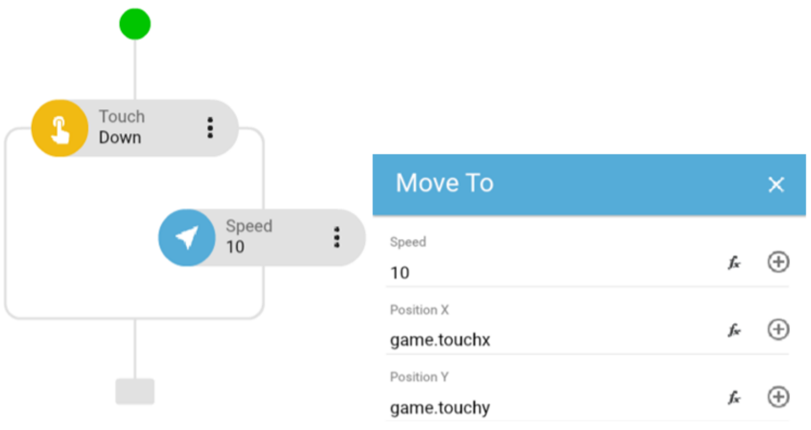
It is known scratch sprites are configured using scripts, having only their position and size as properties. Each time the code is executed, the program enters on a continuous loop. This loop is controlled by two nested conditional statements ‘if’, which are met when the mouse has been clicked and the distance between the object and the mouse is greater than the object size. If that is an affirmative case, the direction vector is computed and the object position coordinates are updated in accordance with the object predetermined velocity and the director vector.

3.3 Gamesonomy’s Game Logic Editor

Continuing with the same example, it is time to test it with the Gamesonomy Game Logic Editor and its flow chart. First, an object or actor is defined that will be the player of the example being programmed. The rule assigned to control the object movement is presented at Figure 5. Besides that, it should be noted that each time an action or condition is added to the Game Logic Editor, a new configuration window is opened to configure the properties of the action or condition.

The action or script assigned to the Player object follows a flowchart structure. Then, the condition under control of the mouse click, called TouchDown, has to be dragged to the graph, positioning it as root of the flowchart. Next, if the condition is met, the action MoveTo has to be dragged to the right branch, that enumerates the developed actions in the case of the root conditions meets. Each action has its own parameters to configure it. On the configuration window for this action, the value for the position of the click coordinates has to be set. They have been stored by the system in variables that Gamesonomy uses by default: game.touchx and game.touchy. Moreover, the velocity of the displacement is set by the programmer.

Only with this short configuration, the game engine updates the position of the object Player by adding the computed displacement on the proper direction.



Move To

Speed: 10

Position X: game.touchx

Position Y: game.touchy

Figure 5. Action programmed with Gamesonomy's Game Logic Editor, and the Move To action configuration window

4. USER EXPERIENCE

Since the intention of this work is to know the real efficacy of Scratch and Gamesonomy after these tools have been used by the students, the direct feedback from them is essential to validate the concept. In order to generate a deeper and nuanced understanding there has conducted a set of questionnaires (Gilchrist, V.J., 1992). For that purpose, an evaluation has been carried out in second year students of the Degree in Video Game Design and Development. The students had been working with these tools for one semester of the first year, and then with a conventional programming language on their second year. This situation will allow determining which one of these two tools is more suitable in order to learn and overcoming a conventional programming subject. The evaluation is based on acceptance test (Davis, F.D., 1989) with the same method used by Zarraonandia et al. (Zarraonandia, T. et al., 2017). This kind of user is ideal for this research: students who in their apprenticeship have to face to conventional programming languages, starting with no background skills (Hanks, K. et al, 2008).

4.1 Objectives and Hypotheses

The models and the learning strategy that use the tools evaluated in this paper are focused on the student, with the purpose to improve the computational thinking and their logical, abstraction and resolution skills. On an educational context, these practices let the students to comprehend how does it works in the real world and empowering them with essential skills to resolve complex problems (Johnson, L.A. et al., 2014).

The aim of the study is evaluate the learning efficacy and the motivational appealing of a visual programming language and a game engine to ease the further apprenticeship on a conventional programming language and to improve their logical thinking. In other words, this work assess the extent to this tools provide students a starting point to start learning programing.

In previous sections, the methodology of programming in visual environments has been explained, specifically through the use of Scratch and Gamesonomy. There is a conviction that this type of programming environments offers clear advantages for the understanding of programming concepts and facilitates the initiation of conventional programming. Based on this situation, with the intention of assessing which of the two tools is more effective, the results of the questionnaires filled in by the students will be analyzed. To this end, a series of objectives and hypotheses were proposed, shown in Table 1.

Table 1. Objectives and hypotheses

Objectives	Hypotheses
O1 - Identify the most effective visual programming tool to develop the student's logical thinking	H1 - Gamesonomy is more effective to develop the logical thinking
	H2 - Scratch is more effective to develop the logical thinking
O2 - Identify the most effective visual programming tool to facilitate the learning of a conventional programming language	H3 - Gamesonomy is more effective to facilitate learning of a conventional programming language.
	H4 - Scratch is more effective to facilitate learning of a conventional programming language
O3 - Identify the most effective tool for acquisition of computational, mathematical and logical concepts	H5 - Gamesonomy is more effective for acquisition of computational, mathematical and logical concepts.
	H6 - Scratch is more effective for acquisition of computational, mathematical and logical concepts

4.2 Protocol

To carry out this study, a sample of 50 students in the second year of the Degree in Video Game Design and Development is used. All students in the sample had used the Scratch visual programming language and Gamesonomy's Game Logic Editor for a semester in their first course. At the end of the second course, and after having taken a conventional programming subject, students were asked to evaluate different aspects of the two tools they had worked in the first semester, in order to collect their opinions.

For this purpose, they have to evaluate the proposed questions on a scale of one to five, with the value 1 corresponding to the lowest level of acceptance of the question, and the value 5 corresponding to the highest level of acceptance. These questions concern the comfort with usability and understanding of the method by specifically addressing to a measure of Perceived Ease-of-Use (PEOU) and Perceived Usefulness (PUSE) (Davis, F.D., 1989) (Davis, F.D., Venkatesh, V., 2004), as it is also attempted to understand the scope in learning of specific programming concepts. The tests were assessed with the average and the standard deviation. It is important to know the statistical significance of the results obtained in the comparison made between Gamesonomy and Scratch. For this purpose, the tests were evaluated with a signed rank based on a two-tailed test with 5% significance Wilcoxon Signed-Rank (Lazar J. et al., 2010).

4.3 Results

To gather information about PUSE and PEOU, this research is based on the questions at Table 2. This test collects information about the learning curve and satisfaction of use related to each of the tools investigated. The survey evaluates questions related to the achievement of concepts related to programming learning.

Table 2. Items and results for the PEOU and PUSE analysis

	Scratch		Gamesonomy	
Questions	Average	SD	Average	SD
PEOU				
Q1 It's easy to learn	3.76	0.73	4.72	0.62
Q2 It is quick to learn	3.67	0.80	3.93	0.78
Q3 It is intuitive	3.62	0.78	4.10	0.64
Q4 It facilitates understanding of conventional programming	4.28	0.73	4.50	0.77
Q5 It improves computational thinking	4.21	0.81	4.52	0.78
Q6 I will continue to program with this tool even though I know how to do it in a conventional programming language	4.12	0.76	4.33	0.46
Q7 I would recommend it to beginners	4.53	0.65	4.70	0.75
PUSE				
Q8 It represents the concept of visual programming	4.80	0.48	4.92	0.63
Q9 It facilitates the understanding of the loop concept	4.31	0.59	4.10	0,59
Q10 It facilitates the understanding of the concept of logic expression	4.21	0.77	4.32	0.63
Q11 It facilitates the learning of mathematical and logical concepts	4.47	0,57	4.71	0.77

Once the students learned and worked at Scratch and Gamesonomy, they realized that their capacity for abstraction and logical thinking had reached the level to learn a conventional programming language more effectively than if they had initially had to face this task without them. This is reflected in a general way in the PEOU test, with the Q4 and Q5 questions of both tools rated with an average of more than 4 out of 5. As for the comparison of these issues among the tools under study, it is worth noting that Gamesonomy is slightly favored, surpassing by 0.22 in the Q4 question and by 0.31 in the Q5 question. As for the measure of perceived ease of use, students clearly preferred Gamesonomy as it is reflected in questions Q1, Q2 and Q3. This would confirm hypotheses H1 and H3.

The results obtained in questions Q6 and Q7, which reflect the general satisfaction of the students with the two tools, should be highlighted. This satisfaction in both cases is above 4 out of 5. Question Q6 demonstrates that both Scratch and Gamesonomy are not considered as simple programming initiation tools, as students confirm their intention to continue using them even after they have learned conventional programming languages. Q7 question demonstrates this satisfaction as they see them as highly recommended for beginners in the world of programming. In both cases, Gamesonomy is slightly above in their assessments.

Looking at the results obtained for Perceived Usefulness (PUSE), the data shows that, except in the Q9 question, Gamesonomy is once again above in the rest of the aspects consulted. The lower acceptance of Gamesonomy in Q9 may be due to the fact that Gamesonomy is a game engine, and there are no loop structures because the system is evaluating all the events in a continuous loop. This means that the student never has to program loops and is therefore unaware that they are programming them and, it would be confirmed hypothesis H6. Q10 and Q11 questions confirmed that Gamesonomy is a better tool for the acquisition of computational, mathematical and logical concepts. This validates the hypothesis H5. Finally, as can be sawn in Q8 question, both Scratch and Gamesonomy have been assessed with values very close to 5, considering them as tools strongly representative of what is known as programming in visual environments, although Gamesonomy is slightly surpassing Scratch.

Concerning the statistical significance of the comparison test between Gamesonomy and Scratch, the Wilcoxon Signed-Rank results establish the test statistic value at 19, minor than the critical value of 73 established for the number of entries. Therefore, it is demonstrated that there is sufficient evidence to suggest that there is a substantial difference between them.

In summary, the results after the analysis of the data reflected in the PEOU test showed that students prefer Gamesonomy as the most effective tool to be used as a basis for computer thought configuration, and as a preliminary step for learning conventional programming languages. Questions related to the PUSE test confirmed that students consider both tools highly representative of visual programming. In addition, Gamesonomy would surpass Scratch as the best tool for acquiring computational, mathematical and logical concepts, while scratch would surpass Gamesonomy as the preferred tool for understanding the loop concept.

5. CONCLUSIONS AND FUTURE WORK

The presented work analyzes the effectiveness of the educational method of teaching programming using a visual programming language or a game engine, in order to facilitate the understanding of conventional programming. In particular, it is used Scratch as visual programming language and Gamesonomy, as game engine. The main aim of this study focused on answer the next question: which one of these two tools is more efficient, appealing, easy to understand and to use for a better transition to the traditional programming? To this end, students have worked with Scratch and Gamesonomy before facing a conventional programming language.

After training and later reflection of the students on the influence of these tools on the understanding of a conventional programming language, the students filled out a test with questions regarding the proposed question. The results of this test have concluded that although both tools are effective and highly representative of visual programming, students prefer Gamesonomy as more effective tool to be used as a basis for computer thought configuration, and as a preliminary step for learning conventional programming languages. The work also confirmed that students have efficiently acquired computational, mathematical and logical concepts with Gamesonomy, but for the understanding of the loop concept they have chosen Scratch.

As future work, we propose to expand this study by adding other tools, and augment the concepts to be studied, such as the use and understanding of complex data structures. It would also be interesting to carry out this study in primary and secondary school students, in order to discover whether this type of learning of programming increases technological vocations among young people.

ACKNOWLEDGEMENT

This work was supported by the Spanish Ministry of Science and Technology (Project TIN2016 - 75866 - C3 - 1 - R).

REFERENCES

Chang, S.K, 1990. *Principles of Visual Programming Systems*. Englewood Cliffs, NJ: Prentice Hall.

Davis, F. D., 1989. *Perceived usefulness, perceived ease of use, and user acceptance of information technology*, MIS Q. 13 (3).

Davis F.D., Venkatesh V., 2004. *Toward preprototype user acceptance testing of new information systems: implications for software project management*. IEEE Transactions on Engineering Management pp 31-46.

GamesonomyTool: URL http://www.gamesonomy.com [Online; Last accessed: 2018-05-19] (May 2018)

Gilchrist, V.J., 1992. *Key informant interviews*. In B. F. Crabtree & W. L. Miller (Eds.), Research methods for primary care, (3), pp 70-89. Thousand Oaks, CA, US: Sage Publications, Inc.

Good, J., 2011. *Learners at the wheel: novice programming environments come of age*. International Journal of People-Oriented Programming, 1(1), pp 1-24.

Gregory J., 2014. *Game Engine Architecture*. Second Edition, 2nd Edition, A. K. Peters, Ltd., Natick, MA, USA.

Grover, S., Pea, R., 2013. *Computational thinking in K–12: A review of the state of the field. Educational Researcher*. 42(1), pp 38-43. http://dx.doi.org/10.3102/0013189X12463051

Hanks, K. et at., 2008. *Sustainable millennials: Attitudes towards sustainability and the material effects of interactive technologies*. Proceedings Conference on Human Factors in Computing Systems. http://dx.doi.org/10.1145/1357054.1357111

Jackson M., 1980. *The Design and Use of Conventional Programming Languages; in Human Interaction with Computers*, pp 321-347; isbn:0126528500; H T Smith & T R Green eds; Academic Press.

Johnson, L.A. et al, 2014. *NMC Horizon Report: K-12* edition. Austin, Texas: The New Media Consortium. Retrieved from http://www.nmc.org/pdf/2014-nmc-horizon-report-he-EN.pdf.

Lazar J. et al., 2010. *Research Methods in Human-Computer Interaction,* Wiley Publishing.

Repenning, A., 2017. *Moving Beyond Syntax: Lessons from 20 Years of Blocks Programing in AgentSheets*. Journal of Visual Languages and Sentient Systems, 68-91. http://dx.doi.org/10.18293/VLSS2017-010

Resnick, M. et al., 2009. *Scratch: Programming for All*. Communications of the ACM, 52(1), 60-67.

ScratchTool: Research on scratch. URL https://scratch.mit.edu/info/research [Online; Last accessed: 2018-05-18] (May 2018)

Van Roy, P. et al., 2003. *The Role of Language Paradigms in Teaching Programming*. SIGCSE Bull. 35(1), 269-270. ISSN: 0097-8418. ACM, New York, USA. http://dx.doi.org/10.1145/792548.611908.

Wing, J., 2006. *Computational thinking*. Communications of the ACM, 49(3), 33-35. http://dx.doi.org/10.1145/1118178.1118215.

Zarraonandia T. et al., 2017. *Using combinatorial creativity to support end-user design of digital games*. Multimedia Tools and Applications 76(6).