Using game-based learning as an approach for teaching kids how to program

Eric Nieuwenhuijsen
Vrije Universiteit Amsterdam
+31653450662
e.c.m.nieuwenhuijsen@student.vu.nl

ABSTRACT
As computers become more and more ubiquitous the need for programmers and the need for comprehending computers becomes bigger. To get people acquainted with computers it is best to start at a young age and in this paper I try to provide some best practices for teaching children how to program and learn computers.

Categories and Subject Descriptors
K.3.2 **[Computer and Information Science Education]**:

Computer Science Education

General Terms
Design, Human Factors, Languages

Keywords
Game based learning, Programming, CS1, Serious games, Educational games, K-12

1. INTRODUCTION
Many researchers have proven that humans learn languages much easier at a younger age than at an older age and this same principle might very well also be valid for learning how to program. Children can be motivated to study simply by providing them with something they consider fun and this is where game-based learning comes in. By giving children the building blocks to create their own games they are playing while learning how to reason in a functional way.

For many now older children, these building blocks may have been LEGO bricks while they were growing up. By playing with these creations you are able to gain spatial visualization ability while simply having fun.

Programming is inherently hard to learn as it includes many concepts people do not actively use in daily life. Some of these concepts include looping, recursion and the use of components to create a bigger whole. Because this is hard to learn it is hard to keep people motivated, adding a gaming and perhaps a social aspect as well could help people in maintaining their motivation when learning how to program.

The main problem with teaching programming is expressed quite clearly by Muratet et al in their 2009 paper[7], they state that "*To be able to program, students need to know programming skills and concepts, but to learn those skills and concepts they have to practice programming*". A proposed approach by Greitzer et al[8] is to "*encourage learners to work immediately on meaningful, realistic tasks.*"

As the IT industries grows and grows learning how to program or at least grasp its basic concepts becomes more valuable than ever. Many parents would like their kids to get this knowledge and as the primary schools do not offer this yet they resort to trying to teach this themselves with varying success. In my paper on this subject I would like to discuss research done on this subject and provide the reader with some current strategies on how to teach this. To conclude this paper I will try and give a recommendation on what now seems to be the general consensus on how to teach programming at a young age.

2. EXISTING WORKS

2.1 Game based learning
Getting precollege students on their way to a computing career has been the subject of many studies since the 90's. The field started out earlier but before the 80's many research was focused on descriptive reports rather than empirical studies.

Early research focused mainly on what students learned but not necessarily on how to best use the media of games to prepare students for more advanced programming and whether this is motivating students to program. This research is mainly about game based learning in general.

Of the earlier works two of the most cited works seem to be from Harel[1] and from Kafai[2]. These works are both mainly based on the Constructionist theory[3] by Papert and Harel. This theory describes how humans learn by making which is exactly what is being done when using games to teach how to program.

In this initial phase of researches the conclusion of many researchers was that the computer is indeed a viable strategy for educating students on a variety of topics. A literature review written by Randal et al.[4] in 1992 concludes that in 68 studies they examined 32% found differences favoring games, 56% found no differences and the other 12% was either questionable or they found a difference favoring conventional instructions.

Of these percentages the percentage gets higher when looking at specific fields of research. For example when looking at math the use of games becomes more superior to traditional classroom instruction and seven out of eight researches they examined were positive. They state that "*subject areas where very specific content can be targeted and objectives precisely defined are more likely to show beneficial effects for gaming*". What's also noteworthy is that they mention that students also reported more interest in simulation and game activities in 12 of the 14 studies where this was researched.

A more recent literature review by Mitchel and Savill (2004)[5] which incorporates the Randal et al. study has the same conclusions in regards to the effectiveness of game based learning. They extended this study by including the more recent studies as well but unfortunately the more recent studies incorporate less empirical research.

An interesting fact they state is that "*It has been found that males are more likely to play to impress friends and for a challenge (Griffiths and Hunt 1995) although girls, too, have been found 'to perceive themselves to have peer approval for moderate amounts of game playing' (Cesarone 1998, page 3)*". This statement could mean that by "exploiting" this need to impress other people could be used as a stimulus to get students to get better results and get better at programming than their peers.

Computer games engage their players and this causes players to keep on playing instead of simply quitting if they fail once. Game designers make players learn the games so well they want to keep playing games and master them. Gee writes the following in his 2003 book[6] about this: "*It is argued that good computer games are not just entertainment but incorporate as many as 36 important learning principles. Taking as long as 100 hours to win, some are very difficult. They encourage the player to try different ways of learning and thinking, which can be experienced as both frustrating and life-enhancing.*"

From empirical research we learn that game based learning can have a positive influence on learning performance but what are the key reasons to use game based learning? Why is this concept gaining in popularity now and not earlier? Perhaps this is due to the fact that we now are able to produce these games but this change can also be motivated by the fact that children grow up with computers and videogames these days.

In his 2006 book[9] Marc Prensky introduces two key reasons:

- "*Our learners have changed radically*"
- "*These learners need to be motivated in new ways.*"

This confirms my earlier assumption as the first reason suggests that "*growing up with digital technology has dramatically changed the way people raised in this time think and process information*" and the second reason suggests that "*the things that were effective in motivating learners in the past do not motivate the learners of today.*"

It seems that the change in culture and the availability of the new technology has caused people to require different stimuli than before.

2.2 Programming and game-based learning

Many languages have been built to allow users to experiment in a graphical way with programming languages. The use of these blocks allows them to not learn any syntax and to start programming right away. I've listed a few examples in the following chapters where this is used.

2.2.1 Starlogo

Starlogo is one of the oldest game based learning frameworks and the first ideas for this are described in a paper from 1994. This is somewhat different than the other frameworks which will follow as it is not strictly a game but due to its age and complexity I have decided to mention it anyway. Starlogo is a "*programming language and environment specifically designed to support simulation design, construction and testing*". This is somewhat different than the other examples but nonetheless interesting because it provides education to students at a different level.

Starlogo is built on, as the name implies, the programming language Logo. The key change the developer made when creating Starlogo was to include parallelism, and this allows you to "*to help non expert users model the workings of decentralized system such as ant colonies*" as said by Mitchel Resnick in his 1996 paper[17].

Users can specify programs for each actor and for each square on the map which all execute in parallel. This allows interactions between many different entities in the system and allows the end user to study massive parallelism without having to learn how to program such hard to create applications. An example from Starlogo can be seen below:

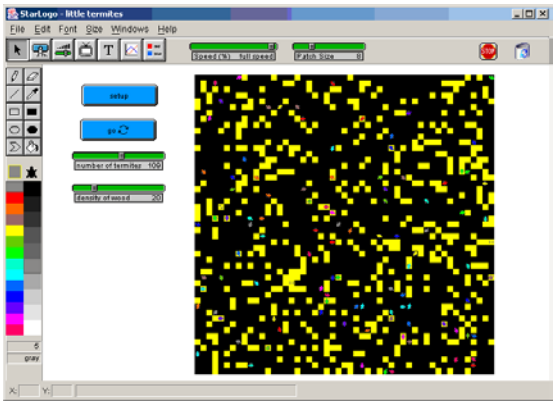

Figure 1 – Starlogo

The large image shows all the different programs running in parallel. By using colors users can see how their changes influence the programs output.

2.2.2 Scratch

Scratch is a name anyone interested in the game based learning field will most likely have heard about. It is one of the longer running research projects and has a large online community. The project has been running since 2003 and since its public launch in May 2007 it has been attracting more and more visitors.

So what is Scratch? It is a graphical programming language that lets children program interactive workflows to develop a sense of systematic thinking. The creators state in their 2009 paper[10] that "*As Scratchers program and share interactive projects, they learn important mathematical and computational concepts, while also learning to think creatively, reason systematically, and work collaboratively*". This does not mean that it is used purely to get children ready for being a programmer later but it helps them in all sorts of jobs and programming is just one of them. It helps children getting used to use certain constructs to express ideas.

An example workflow in Scratch could look like this:

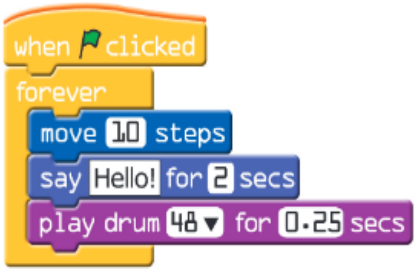

Figure 2 - Scratch example

These simple blocks can be used by children to create all sorts of interactive storyboards. By thinking in loops and in small steps children are forced to use a systematic approach to get the story they want on the computer. Another nice thing about Scratch is

that the sharing part is very easy; this is something very important as it allows children to share their creations with others and get ideas and feedback from others.

An example case of this is presented in the 2008 paper[11] of Maloney et al. called "Programming by Choice". This paper describes a case study at a Computer Clubhouse where children aged 8-18 actually favor using an environment such as Scratch over other more common games like playing on an Xbox. In their study they also found that more than 50% of the projects included some sort of looping construct, other popular topics included user interaction, conditional statements, and communications and synchronization.

The writers attribute this success partially to the fact that "*Most youth didn't identify scripting in Scratch as a form of programming*". They simply saw it as something "cool" and as a part of the culture which engaged them more socially. In their paper they also quote Kelleher and Pausch[12] who state that "*by simplifying the mechanics of programming, by providing support for learners, and by providing students with motivation to learn to program*". All three areas described here are actually addressed by Scratch they argue.

2.2.3 Alice2

Alice2 is similar to Scratch in a way that they both use a block style interface to control a graphical output. By using a drag and drop system users can create programming structures as taught in programming classes.

In their 2002 paper[13] the authors of Alice2 state that "*Beginners must learn to find structured solutions to problems, express those solutions in a rigid, formal syntax they must memorize and mechanically enter, and learn to understand the behavior of the running program*".

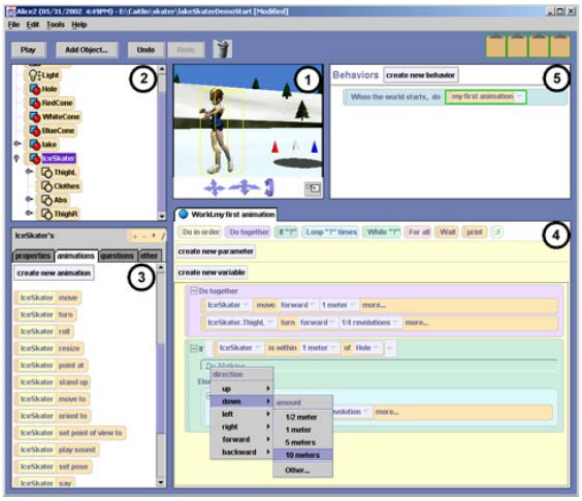

Figure 3 - Alice2

Alice2 seems to be comparable to scratch but there is one key factor missing in their design and that's the social / sharing aspect. Whereas Scratch is really focused on showing off your creations and getting help/suggestions from other users Alice2 seems to be more for the user that wants to do it on his own.

2.2.4 Examples from education

Another option for game based learning is to incorporate this in an actual video game and to use an element of competition to motivate people. By letting people battle each other using some sort of artificial intelligence you can motivate students to create a better bot because they want to win.

An example of this was in the VU Bachelor Course "Intelligente Systemen" (English: Intelligent Systems) and the course "Pervasive Computing". Both of these courses had a competitive element included which helped in motivating students to try and win even though there was no large money prize to be won. In both courses you were teamed up with a fellow student which also had a positive influence on the end result by allowing discussion.

The first course, Intelligent Systems, was based around the game Planet Wars. This is a game from the Google AI challenge in 2010 and it was simplified for the course to make it easier for students to participate and get an actual working bot. The game is a strategy game set in outer space and your objective is to take over all planets on the map or to eliminate all of your opponent's ships.

By providing students with a Java starter package and some very simple bots users can start by experimenting with modifying these existing bots. This learning by modifying and simply trying helps but it does require that you know some basic Java already. By simply trying and playing against your own bot you can test its performance and intuitively improve the performance by adjusting cases where the bot fails to perform properly. This process encourages students to think of problems such as the space-time tradeoff and also to rehearse their Java skills because they actually have to program.

The second course, Pervasive Computing, did not include any actual syntax but used a slightly different approach. By using a block style software called RoboPAL students had to make a LEGO Mindstorms robot do all sorts of assignments. By letting students experiment using a simple visual language it becomes easier to learn things such as constructs or recursion.

This form of education is also suited for younger children; because everything is displayed in a visual way everyone that can read is able to direct the robot. The education material used the practical part of this course was actually the same material used for teaching 15-17 year old children how to program robots. A simple example program can be seen below.

Figure 4 - RoboPAL

This program simply sends the robot back and forth over a straight piece of land. By allowing children to program simple programs such as this into a simulator or even into an actual robot you learn how to think sequentially and you can see what happens when the robot branches in your program. This prepares users for the concepts used in actual programming.

3. CURRENT STATUS OF RESEARCH

A rather recent tool for learning programming via game based learning is called Pex4Fun, this is a tool released by Microsoft and described by some of their employees in a paper[14] from 2013. This tool takes a different approach than the earlier mentioned tools because it focuses on writing actual code and you can win the game by writing functions that adhere to the specification.

The problem with writing actual code is that you do need some way to validate this. In Massive Open Online Courses (MOOCs) this is often done by peer grading but this might not have the wanted accuracy. In Pex4Fun Microsoft used an automated test suite to test whether your code is correct. This allows users to quickly try multiple solutions and immediately get feedback on whether their solutions are correct.

This approach can be used to train introductory programming courses by starting off with some short syntax training and then letting people user their account to do some of the exercises. They vary from very easy (i.e. calculate the square of 21) to harder ones (i.e. cipher codes). Although this is not a graphical solution I feel that a solution like this would work well for students.

In this paper they note the social part and state that "*Holding a contest of solving coding duels in either a public setting or a classroom setting can serve the purpose of engaging students to solve coding duels in a dynamic social context within a specific period of time*". This challenge aspect seems key in motivating users to win and get the best solutions possible.

An interesting problem next to the solution for game-based learning is the acceptance of the teachers. Teachers can sometimes be reluctant to switch their lesson plan to something digital which they might not fully comprehend because they didn't grow up with computers. In their 2013 paper[15] Bourgonjon et al describe a case study where they asked over 500 teachers what their opinion is on using a system like this.

Their research shows that "*On the one hand, teachers are not really convinced that video games are very useful for enhancing their job performance. On the other hand, they believe that video games provide opportunities for learning in a similar way that teachers perceive the merits of ICT in the classroom*". This translates into the fact that teachers do not intend to use games in the near feature but when offered to use them most likely will use them.

The root of this acceptance problem might lie at the fact that teachers have a limited frame of reference and when thinking of games immediately think of commercial games which might cause this negative influence. Another problem is that there are simply not enough people who are skilled in programming enough to teach this to other people. One possible solution for this is to not let the teachers give these courses as it will be hard to change this idea.

A possible solution is provided by Peter Greuenbaum in his 2014 article[16]. He suggests that undergraduates teach programming to middle school students using the earlier mentioned language Scratch. The major benefit of doing this is that there is no need to educate teachers with programming and that this relieves the stress for older teachers that have to use a computer to teach. The added benefit for the undergraduates is that they get a crash course on how to teach a class of middle school students.

The undergraduates were given a quick course on how to use Scratch and where then tasked with developing a lesson plan in a group. This lesson plan was finalized and after this they were tasked with getting interest from the middle school students. By organizing an activity at a school Halloween party they recruited students to participate in the course.

Over the course of three sessions the children were taught some basic programming principles which included conditionals, loops and events. The nice thing about this research is that all the children got to choose their own images; this ensured a unique look and feel for each game and gave them the experience that they were creating something of their own.

In the conclusion of the research the author mentions that "*Using undergraduate computer science majors to teach middle school students is an effective way to bring computing expertise into learning environments where very few teachers have the skills to teach this subject*". This is a valuable conclusion as using these undergraduates can help relief the problems teachers might have with teaching like this as it can have undergraduates take over so the teachers do not have to teach subjects they might not be very good at.

4. CONCLUSION

Unfortunately it has proven to be hard to find actual research done in teaching programming to children under 18. Most of the research found in papers focuses on university students rather than earlier. In this paper I have provided the reader with a few examples where this research was indeed done in the designated age group. Based on the other examples I have created a few general recommendations for teachers or researchers that want to set up a program to teach younger children how to program.

The first and in my eyes most important recommendation I would like to give is to make your application social. Children should be able to pair up or share creations with each other to help each other and spark ideas. The helping of others will train both the person being helped and the helper and increase their knowledge on the subject. Sharing experiences and finished products is important for ideas and help. The social part helps as it lets students motivate each other while also providing support.

Another important point is that when teaching the basics of programming you shouldn't have to be too focused on syntax. It is not uncommon that introductory programming courses start off in a Java IDE and students will be instructed to start with things like outputting "Hello world!". Although this is a good start to learn a specific programming language this does not contribute to the student grasping the actual concepts of programming. The fundamental understanding of how programming works is key in becoming a programmer that can program in any language.

For younger children it is especially important that they do not see the programming part as a chore or as a school thing. If it is possible to include it in some sort of game children will see it as something cool and might actually do this instead of picking up that Xbox controller and playing a commercial game.

The actions done by the users should be made as visual as possible. By providing a visual interface children will be attracted to the visual part more and will be able to experience their creativity in using fun and inviting graphics for their game. The choice of graphics will give the program a personal touch and masks the programming part as this will then become more of a step in creating the graphical experience that they want.

5. FURTHER READING

As a first reading suggestion I would highly recommend the article about teaching middle school children using undergraduates [16]. This article is a good example of how combining university students with middle school students can generate positive effects for both groups and provides an excellent idea for more research into this topic. A study like this could perhaps be executed on a larger scale with more students and a more scientific way of measuring the effects. It would be nice to also see if they are actually better at programming later on.

Another interesting topic I came across on the internet is the recently funded Kickstarter project Hello Ruby[20]. The author aims to publish a book meant for 5-7 year old children with which they can learn to program. The author describes the book as "*a children's book that teaches programming fundamentals through stories and kid-friendly activities*". Now that the Kickstarter has ended there is also a website which you can see at [19].

While browsing the internet for more information on game based learning I also found the following not entirely scientific but nonetheless interesting article on gamification [21]. This focuses on the training departments and how you could improve your learning route by gamifying your existing trainings.

For the readers that know how to program I would highly suggest taking a look at Microsoft's Pex4Fun program [18]. You can login using a windows live account and something like this is a good way to train your language syntax while also practicing your ability to solve problems and deduct solutions from output.

6. REFERENCES

[1] Harel, Idit. *Children designers: Interdisciplinary constructions for learning and knowing mathematics in a computer-rich school*. Greenwood Publishing Group, 1991.

[2] Kafai, Yasmin B. *Minds in play: Computer game design as a context for children's learning*. Routledge, 1995.

[3] Papert, Seymour, and Idit Harel. "Situating constructionism." *Constructionism* 36 (1991): 1-11.

[4] Randel, Josephine M., et al. "The effectiveness of games for educational purposes: A review of recent research." *Simulation & Gaming* 23.3 (1992): 261-276.

[5] Mitchell, Alice, and Carol Savill-Smith. "The use of computer and video games for learning: A review of the literature." (2004).

[6] Gee, James Paul. "What video games have to teach us about learning and literacy." *Computers in Entertainment (CIE)* 1.1 (2003): 20-20.

[7] Muratet, Mathieu, et al. "Towards a serious game to help students learn computer programming." *International Journal of Computer Games Technology* 2009 (2009): 3.

[8] Greitzer, Frank L., Olga Anna Kuchar, and Kristy Huston. "Cognitive science implications for enhancing training effectiveness in a serious gaming context." *Journal on Educational Resources in Computing (JERIC)* 7.3 (2007): 2.

[9] Prensky, Marc. "Computer games and learning: Digital game-based learning." *Handbook of computer game studies* 18 (2005): 97-122.

[10] Resnick, Mitchel, et al. "Scratch: programming for all." *Communications of the ACM* 52.11 (2009): 60-67.

[11] Maloney, John H., et al. "Programming by choice: urban youth learning programming with scratch." *ACM SIGCSE Bulletin*. Vol. 40. No. 1. ACM, 2008.

[12] Kelleher, Caitlin, and Randy Pausch. "Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers." *ACM Computing Surveys (CSUR)* 37.2 (2005): 83-137.

[13] Kelleher, Caitlin, et al. "Alice2: programming without syntax errors." *User Interface Software and Technology*. 2002.

[14] Tillmann, Nikolai, et al. "Teaching and learning programming and software engineering via interactive gaming." *Software Engineering (ICSE), 2013 35th International Conference on*. IEEE, 2013.

[15] Bourgonjon, Jeroen, et al. "Acceptance of game-based learning by secondary school teachers." *Computers & Education* 67 (2013): 21-35.

[16] Gruenbaum, Peter. "Undergraduates Teach Game Programming Using Scratch." *Computer* 47.2 (2014): 82-84.

[17] Resnick, Mitchel. "StarLogo: An environment for decentralized modeling and decentralized thinking." *Conference companion on Human factors in computing systems*. ACM, 1996.

7. LINKS

[18] Pex4Fun, http://www.pexforfun.com

[19] Hello Ruby Website, http://www.helloruby.com/

[20] Hello Ruby Kickstarter, https://www.kickstarter.com/projects/lindaliukas/hello-ruby

[21] Article on gamification, http://www.learningsolutionsmag.com/articles/1337/gamification-game-based-learning-serious-games-any-difference

Why should games be used in classrooms? â€œGames are effective tools for learning because they offer students a hypothetical environment in which they can explore alternative decisions without the risk of failure. Thought and action are combined into purposeful behavior to accomplish a goal. Playing games teaches us how to strategize, to consider alternatives, and to think flexiblyâ€ (Martinson and Chu 2008: 478). That quote summarizes my beliefs about using games to teach, practice and reinforce a foreign language.Â Games allow the students to work as a team and to work collaboratively towards a common goal. This collaborative effort is more than just learning to work with others. It promotes a symbiotic relationship where they can learn from each other. Learning plan. The first time I participated in the program, our group taught kids rather mindlessly: we were coming up with simple tasks to explain different operators. By the end of the course we had nothing concrete to evaluate, analyze, and share. This second time I decided we are going to create a memory game with kids.Â We used Scratch as our development tool. Scratch is a great tool to teach kids to program because each action, each operation is represented graphically. For example, you can rotate a cat 360 degrees in 1 second using the following script: Here's how it looks like in actionÂ Teachers finally have time to discuss the lesson: discuss the kids, approaches to shy and active kids, plan next lessons. We had the following decisions Game-based learning is a teaching method that allows learners to explore different parts of games as a form of learning. Games can be designed by teachers and other education specialists in a way that balances academic subjects such as history with the strategies, rules and social aspects of playing a game. As a side-effect of technological growth, game-based learning often generates negative connotations because of its close association with video games, which inevitably raises questions about its consequences. However, these games are typically designed at different ability levels and with t Classroom education doesnâ€™t have to be mind-numbing! Use game-based learning in the classroom to help engage students in the learning process.Â How to choose the right games for your classroom. If youâ€™re teaching students about the wonders of cartoonish architectural warfare, perhaps Fortnite may be relevant in your classroom. But otherwise, not so much. Obviously, the games you choose to include in your lessons need to align with your teaching goals.Â What kid doesnâ€™t want to play as a Warlock rocking elder robes and spells while also learning about the structure and function of DNA? The best classroom games allow students to learn the material while giving them the creative freedom to express themselves in their own unique ways. Purpose.