

SILVER
CODERS

IO1-METHODOLOGICAL DIGITAL LEARNING FRAMEWORK

Document Info	
Project reference	2020-1-SE01-KA227-ADU-092582
Intellectual output / Activity	IO1-METHODOLOGICAL DIGITAL LEARNING FRAMEWORK
Dissemination level	Global
Date	22/10/2021
Document version	1.0
Status	Final
Authors	Virtual Campus Lda / FU-UPPSALA
Reviewer	All partners
Contributors	All partners
Approved by	Steering Committee



CONTENTS

INTRODUCTION	5
RESEARCH	7
O1/A2 - Stakeholder mapping	7
Stakeholders	7
Stakeholders involvement in the project	11
Examples of individual stakeholders' organizations	12
O1/A3 - The current status quo in programming skill development in adult education	14
Tools and programs available for adults	15
O1/A4 - Current trends in digital tools for building programming skills	19
Visual Programming Platforms	19
Scratch	20
Snap!	22
Alice	23
Tynker	25
App Inventor	27
No-code Platforms	30
Airtable	31
Appy Pie	34
Bubble	36
Gamified Programming Platforms	38
Code Combat	42

Human Resource Machine	43
LightBot	44
May's Journey	45
No Bug's Snack Bar	47
Robot ON!	48
Educational Pacman Game	49
CMX	50
Low-Code Game Editors	51
GameMaker Studio 2	51
RPG MAKER	53
CONSTRUCT	55
GDEVELOP	55
STENCYL	57
O1/A5 - Programming skill building requirements for adult education	58
Expected general digital competencies	59
Expected programming related competencies	62
O1/A6 - Skill development requirements for adult trainers	65
SILVERCODERS LEARNING FRAMEWORK	70
Requirements' analysis	70
Learning Strategy	76
REFERENCES	78

INTRODUCTION

One of the main objectives of the Digital Skills and Jobs Coalition, as defined by the European Union, is to ensure that everyone has the right digital skills to thrive in society and on the labour market. However, this demand is not aligned with the existing offer as there is a major shortage of qualified staff. In this scenario, there is a need to improve the citizens' skills, especially older adults' competencies, since their digital exclusion prevents them from being fully integrated into the Knowledge and Information Society. This exclusion became even more prominent in the context of the COVID-19 pandemic crisis, which exposed some groups' difficulty in adapting to the new societal challenges. In particular, it is important to focus on coding and programming skills, since “coding is today's literacy, improving skills like problem-solving, teamwork, and analytical thinking and enhancing creativity, teaching people to cooperate across physical and geographical boundaries and to communicate in a universal language (DG Connect, 2020).

The SILVERCODERS (Developing the Creativity of Older Adults through Coding) project intends to develop trainers and adult learners' digital and creative abilities by engaging institutions and organisations in formal, informal, and non-formal education for adults with companies from the creative sector. Doing so intends to provide adults with the necessary tools and competencies to develop creative and innovative solutions to face new risks and challenges, both in personal, educational and professional contexts. This can be important in all sectors of activity, but even more, in the creative and cultural sectors, which could benefit from becoming more digital and modern because this renovation would contribute to making the sector (one of the most hardest-hit ones) more adaptable, resilient, and able to survive and prosper in the current situation and possible future challenges as well.

The project's Intellectual Output 1 aimed at developing a sound methodological learning framework for building programming and coding skills among older adults. To do it, the organized activities involved people working in creative fields, such as programmers,

designers, and content producers to combine their digital skills and creativity with the following goals:

- A) Work related to learning requirements definitions to analyse the existing situation.
- B) Work related to the design of the SILVERCODERS learning methodology for building programming skills, for instilling user-centered mindsets, and for promoting creative and innovative thinking.

Specifically, the tasks included in this Output were the following:

- Identify the stakeholders in adult education that stand to gain from the SILVERCODERS proposed learning methodologies and tools and document their needs and characteristics. The work analysed the needs of adults, trainers, and other potential stakeholders.
- Identify the current situation in adult education about teaching programming and document current practices.
- Document the level of deployment of digital learning tools in the context of digital skill development.
- Document other tools and services related to the development of programming skills.
- Analyse the skill building needs for adult trainers on promoting programming skills and positive attitudes among adults in relation to the creative sector.
- Design feedback mechanisms that allow learners to build knowledge from the response of the system (and the teacher) to their efforts.
- Design the user interface of the learning tools, i.e. the way in which learner will interact with the tools.

The result was a learning methodological framework that enhances existing practices on programming skills development.

RESEARCH

To be able to reach the learning framework for enhancing programming skills in adult education the following set of activities were conducted:

- O1/A2 - Stakeholder mapping
- O1/A3 - The current status quo in programming skill development in adult education
- O1/A4 - Current trends in the deployment of digital tools for building programming skills
- O1/A5 - Programming skill building requirements for adult education
- O1/A6 - Skill development requirements for adult trainers

O1/A2 - STAKEHOLDER MAPPING

The objective of this activity was the analysis of a precise map of stakeholder groups that stand to benefit directly or indirectly from innovative pedagogical interventions that promote programming skill development in line with community and real-world needs.

Stakeholders

The COVID 19 pandemic underlined the urgency to work on the digital divide. The coronavirus crisis has accelerated the uptake of digital solutions, tools, and services, speeding up the global transition towards a digital economy. However, it has also exposed the wide division between the connected and the unconnected, revealing just how far behind many are on digital uptake. Inequalities in digital readiness hamper the ability of large parts of the world to take advantage of technologies that help us cope with the coronavirus pandemic by staying at home. SILVERCODERS contributes to tackling this issue, by developing the digital and creative abilities of 55+ adults and equipping them with programming and coding skills.

To achieve this aim, the direct stakeholders will be:

- **Trainers, adult educators or specialist staff**
 - will develop new pedagogical and ICT skills to promote and teach the use of digital training tools in their training practices;

- the role and status of trainer will acquire a new current dimension in this society of knowledge and digitalization by increasing ICT skills and literacy;
- will develop new creative skills and inspiration for training and other working methods;
- will improve their training and support offer on the creation and digitization side using coding;
- will become more competitive on the trainers market, being able to adapt the SILVERCODERS methodology and create new course curricula for other potential beneficiaries;
- will be able to use the educational resources of SILVERCODERS, but on the other hand they will have the possibility to adapt them (multimedia contents, documents, guidelines, and tools) to another target group, which will bring them an added value and trust.

2. Adult learners

- will increase their digital skills and literacy;
- will understand the concepts of programming and coding and being able to create apps and becoming more tenderers for employers;
- will reinforce their cognitive and creative skills;
- will improve their employability skills.

3. Organizations connected to adult training and well-being (Research institutions in adult education, adult or senior associations, institutions or organizations dealing with adult vocational training, lifelong learning departments in local, regional or national councils, the department of innovation and lifelong learning in within the ministries of education etc)

- will be able to provide innovative solutions for enhancing the use of ICT by these adults.
- will have an increased awareness on problems faced by trainers and adults regarding the use of digital devices and how they can be improved

- will have an increased knowledge of innovative experiences and practices carried out in other countries that could be of inspiration for new policies and services.

All these groups, as well as local stakeholders and communities will also benefit from:

- Increased openness towards other European countries and cultures
- Increase their networking capacities and visibility at national and European level
- Increased awareness about the contribution of the European Union activities towards social inclusion and equal opportunities

Indirectly, there will be a set of different organizations that, by their nature, will also become relevant stakeholders:

- **Adult training organizations (ATO):** Synergies with training organization focused on adult learners 55+ should be searched, to involve potential participants in the training paths we will offer and to learn from already taken practices. Particular interest should be given to the international experience of the University of the third age, a nationwide network of learning groups aimed at encouraging older people to share their knowledge, skills, and interests in a friendly environment.
- **Territorial business associations (TBA):** Through their involvement in the project, it will be possible to get in touch with the companies of the creative and cultural sectors, collect contacts and create links with them. Other companies from relevant sectors might be involved as well.
- **Local public administrations and policymakers (LAs):** The involvement of the public institution at a local level will guarantee impact and sustainability to the project.
- **Senior informal associations (SIA):** Several kinds of elderly people associations, such as the senior centres, represent potential bodies to target for involving 55+ seniors.
- **NGOs, foundations, and associations (NGO):** These are entities that might have started other projects related to the same topic, in the same territory. They should be included among our stakeholders, for the possibility to exchange experience and possibly coordinate the intervention.

- **Formal and informal caregivers (FIC):** The caregivers of older persons, both formal and informal, are precious stakeholders: they could be both targeted by the project and a way to find other seniors to involve in our action, as they represent an important link to the seniors.

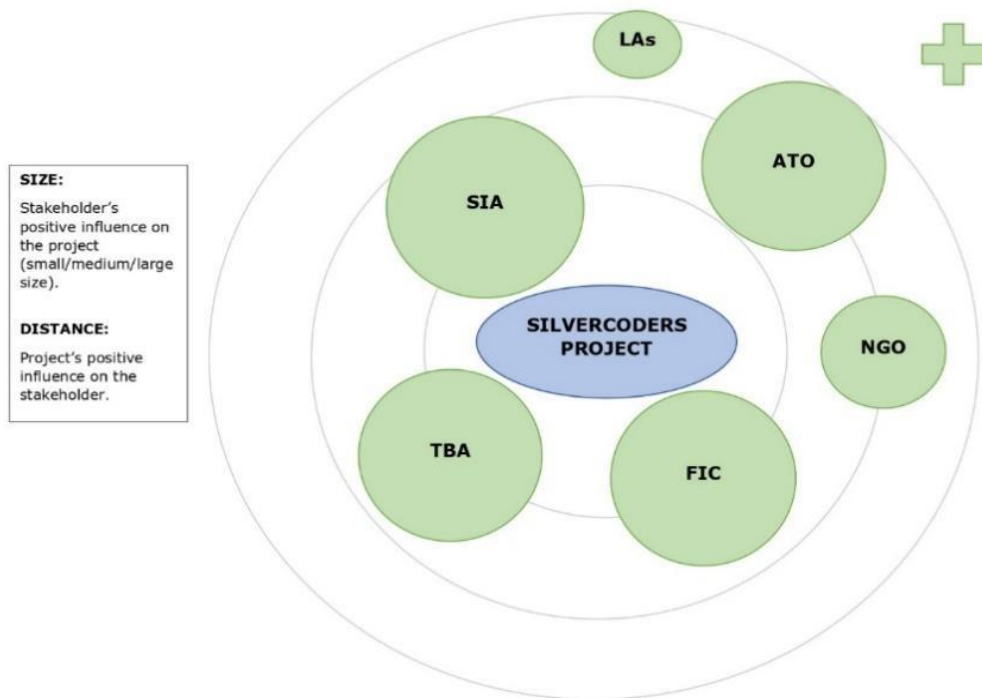


Figure 1 SILVERCODERS Stakeholders

As noticed from the map, we considered SIA (Senior informal associations), TBA (Territorial business associations), FIC (Formal and informal caregivers) and ATO (Adult training organizations) to have a strong influence on the project, therefore they are represented by a large bubble. SIA and FIC are two categories of stakeholders that will be crucial in finding 55+ adults to involve in our project, while TBA will be our contacts for involving companies in the creative and cultural sectors. The three categories are placed close to the project's bubble because of the positive influence of the project itself on them.

ATO share the same level of influence on the project as SIA, TBA and FIC, but they have been placed farther from the project's bubble because of the fewer influence SILVERCODERS will have on the work of a training organization.

NGO (NGOs, foundations, and associations) size and distance reflect their middle degree of influence on the project, and vice versa: we considered the impact of positive synergies and coordination, which is mainly indirect.

LAs (Local public administrations and policymakers) still influence stakeholders, but both the influence they have on the project and the one the project has on them has been considered as low: that led to a small bubble, far from the project's one.

Stakeholders involvement in the project

A successful dissemination strategy should be based on the understanding of stakeholders and their information needs and preferences. The purpose of the activity should be based on engaging, informing, promoting the project outputs. To ensure that the project results will be used, the dissemination activities, within the framework of a broader plan, will be particularly focused on developing strategies to reach out stakeholders, relevant institutions, organizations, and individuals.

The project expects to reach directly, in the training stages, about 100 trainers and 350 adults. The multiplier events will be the opportunity to reach at least another 210 stakeholders. Through other dissemination actions, the project expects to reach another 3500 adults.

Defining the purpose of dissemination is a first step to decide on the audience, message, method, and timing of the dissemination. Doubtless, there are a wide variety of dissemination methods and tools. SILVERCODERS will, accordingly select the right ones to address the target audience and achieve your purpose.

The main aim of the multiplier events will be to disseminate the intellectual outputs. SILVERCODERS dissemination strategy is based upon a set of tools to:

- **built up a network:** a mailing list of individuals interested in being involved and informed;

- **promote the project and relevant outputs:** press releases, newsletters and flyers aimed at create awareness about the project and translated to national languages of the involved partners;
- **social media, blog posts, reports, journal articles** can be used to reach a wider audience;
- **events, conference, workshops** promote the exploitation of the main project results engaging the direct and indirect target group and stakeholders at local, regional, national and European level.

Examples of individual stakeholders' organizations

- The Institute of Educational Sciences (ISE), Bucharest, Romania, is a national institution for research, development, innovation and training in the fields of education and youth, which actively contributes to innovation in education through expertise, training, studies and research. ISE's mission is to provide the necessary scientific support for the latest approaches in education, for authentic, motivating, active and creative learning.
- Unieda, Unione Italiana per l'Educazione degli adulti. Italian Association for the Education of Adults
- Cnupi, Confederazione Italiana delle Università Popolari Italian Confederation of Folks Universities (Università Popolari)
- ANOP, the Associação Nacional de Oficinas de Projectos - Desenvolvimento e Educação which is a private non-profit association that was created in 1999 by a group of organisations concerned with local development, vocational training and job creation. It is made up of professionals with experience mainly in the areas of adult education and training. Its headquarters are in Santa Maria da Feira, in the Northern region of Portugal.
- ENTRE-SERRAS (Associação de Desenvolvimento do Concelho de Pampilhosa da Serra). As a non-profit association it seeks to promote the development of the Pampilhosa da Serra's region. This target is pursued through a holistic approach which involves work in various fields. The efforts to foster a culture of partnership in Education.

- Popular Universities (Universidades Populares), associations, and centres for social initiative. Some of them are focused on groups of disadvantaged people, such as the elderly, women, immigrants, or disabled people.
- Instituto Paulo Freire de España its activity is related to the non-formal learning, with a specific focus on adult education. It is a network of organisations and people involved in the promotion of adult learning but also engaged in the promotion of social inclusion and in combating racism and discrimination. It provides courses and seminars in order to promote the awareness of adult education issues in Spain and makes publications on these themes.
- ACEFIR, the Catalan Association for Education, Training and Research, manages a social initiative that brings together a team of professionals from different fields with the common interest of working for education, training and research related to youth, adults and the elderly.
- Associació ESPIRAL - Entitat de Serveis, is a private non-profit NGO that since 1992 has been working for the integral development of individuals and the social cohesion of local communities. It comprises professionals from the fields of education, health and social services. Espiral supports organisations to provide a European dimension to local projects and promotes good practices. It also provides support and expertise for the implementation and monitoring of projects.
- The UPDEA Foundation is a non-profit, private and independent entity, founded in Madrid in 1999, whose purpose is to contribute to the cultural, social and personal development of adults through lifelong learning.
- Regional Vocational Training Centers (KEKs) are activated in the field of non formal training, certified to provide training opportunities for those who are disadvantaged. The Vocational Training Centre "DAFNI" was founded in 1996 by a team of scientists with the goal to combat unemployment (especially in women and young people) with high-level vocational training based on the latest educational methods and technological innovations.

- The ERGON KEK (centre for Vocational Training) was founded in 1995 and it is one of the first Centres for Vocational Training that operated on a nationwide level. Since then, it has established itself as one of the most dynamic businesses in the field, having acquired a great deal of experience by providing vocational training and learning services.
- The Hellenic Adult Education Association (HAEA) is a non-profit, nongovernmental organisation. The mission of the association is to promote the scientific development of adult education in Greece and South Eastern Europe, to support its members by providing professional development, enhancing communication, and creating a sense of community

01/A3 - THE CURRENT STATUS QUO IN PROGRAMMING SKILL DEVELOPMENT IN ADULT EDUCATION

To get an understanding of the current status quo in available tools and programs for programming skills development for older adults, a desk research was conducted. The aim was to identify any already existing practices, both formal and non-formal, taking place at National (in each country represented by the consortium), International and European level.

The desk research clearly showed that as of today, there is great awareness of the importance and benefits of coding and programming skills in order to thrive in society and on the labour market, where such tools are fundamental for professional, personal and social tasks. Additionally, “Coding enhances creativity, teaches people to cooperate, to work together across physical and geographical boundaries and to communicate in a universal language.” (European Commission, 2014). In this perspective, at a National, European and International level, there are many initiatives being carried out to ensure that the general population, and especially youth and young children, have access to information, resources and training from an early age. This applies to both formal and non-formal settings; there are many projects and initiatives aimed at empowering teachers to include these skills in the classroom, as well as others that focus on youth being able to further develop coding and programming skills in their leisure time, even from an early age.

Tools and programs available for adults

At a national level, there are hardly any initiatives aimed at adults, much less +55 adults.

In Portugal, one of the initiatives being carried out is “Programa Seniores Ativos” (Active Seniors program), promoted by the City Hall of Sintra. Its aims at promoting the well-being of elderly people and their careers, through online classes. The initiative focuses on areas such as: new technologies, programming and coding, music, arts and science, as well as many others.

In terms of initiatives focusing exclusively on programming and digital skills, there is “Programming for everyone!”, an initiative developed by the Information sector of the European Commission in Portugal, together with the European parliament. This project created two online courses for inexperienced people in areas regarding basic concepts of programming, web design and data analytics.

In the other countries of the consortium, it wasn’t possible to find any specific initiatives aimed at 55+ adults. Most initiatives that already exist are mainly focused on activities for older adults in general, where programming skills are also available, or aimed at developing digital and programming skills for adults in general, without a specific age group.

However, at a European level, there are some good practices focused on programming and coding that could be of relevance to the task at hand. Two initiatives that were particularly interesting are: EU Code Week and The European Coding Initiative, also known as ‘all you need is {C<3DE}’.

The “EU Code Week” is a grassroots initiative run by volunteers and supported by the European Commission as part of its strategy for a Digital Single Market. Its objective “is to make programming more visible, to show young, adults and elderly how you bring ideas to life with code, to demystify these skills and bring motivated people together to learn” (European Commission, 2014). Apart from organizing the EU Code Week, where in 2019, 4,2 million

people participated, involving more than 80 countries around the world, they have activities and resources available for anyone who wants to get into coding.



Figure 2 CodeWeek

The European Coding Initiative goal is to promote coding and computational thinking at all levels of education, as well as in more informal settings. Its aim is to promote coding through a mixture of online and offline, real-life activities, with the intention of establishing coding as a key competence within every education system in Europe. Their website provides resources ranging from ones aimed at children, to pedagogical resources and lesson plans for teachers, to industry training and certification for professionals.

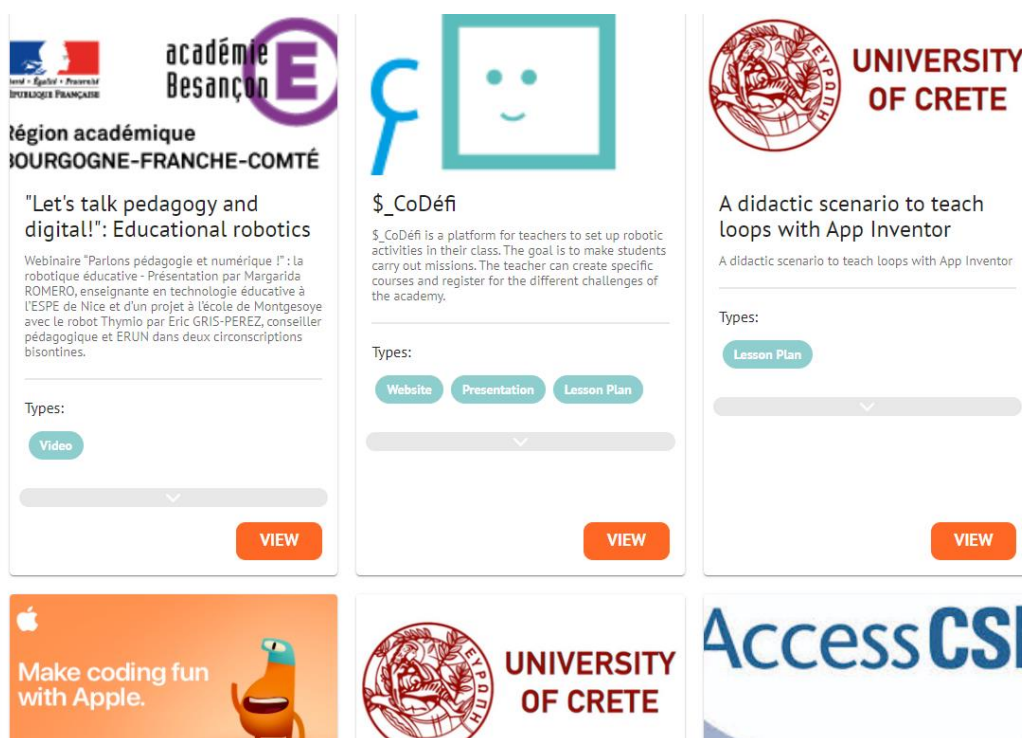


Figure 3 European Coding

There is one Erasmus+ project, called Silvercode, that has the same goal and target groups.



Figure 4 Silvercode project

As stated on their website, “the project objective is to develop digital literacy for elder citizens and especially learning basics on how to program”. Its target group is the elder EU citizens, and they have developed a training course available on their website, as well as a “Silver Coding” community-based platform, forum and social network groups. During the project's course, they had peer-to-peer events where trained elders trained fellow elders, to introduce them to the basics of coding.

The training course consists of six different modules, each with a different theme. The registration for this course is free of charge and anyone who wished to participate in the piloting phase was welcome to do so. The aim of the course is to help the learner improve their overall digital knowledge and skills, to develop computational thinking and stop being intimidated by strange words and unknown concepts like coding and many others. After each unit, they included a glossary so the learner gets a better understanding of the glossary used within coding. For their project, computational thinking is at the heart of the learning that they advocate. It is the thinking process that underpins computing and digital making: formulating a problem and expressing its solution in such a way that the computer can effectively carry it out. Computational thinking covers knowledge and skills including, but not limited to logical reasoning, algorithmic thinking, pattern recognition, abstraction, decomposition, debugging, problem solving (Silvercode, 2021).

In short, these are the initiatives that were found available at European level. Only one of them is specifically aimed at older adults; in the other two, although they include people of all ages, they do not have specific objectives or resources for our target group.

The situation on a more global and international level is very similar regarding programming skills development in adult education. One important aspect to be mentioned here is that there is a considerable amount of free (and paid) resources for adults available on the Internet, with a clear orientation towards self-learning.

In this sense, there are still no resources for adults over 55 specifically, but there are many tutorials, courses, communities, and many other tools, available for the development of

coding skills from scratch, or to continue developing or updating the skills they already have. Especially when it comes to the development of coding skills with no previous knowledge, a lot of these initiatives use elements of gamification, to keep participants engaged and motivated.

As an overall summary of this research, we can conclude that there are almost no projects or initiatives at National and European level dedicated to developing the programming skills of adults, less so dedicated exclusively to adults over 55. Most of them target children and youth, and the ones that do include adults are usually aimed at the global population. There are a lot of free resources for self-learning for adults available on the internet, but, once again, nothing specific for the projects main target group.

O1/A4 - CURRENT TRENDS IN DIGITAL TOOLS FOR BUILDING PROGRAMMING SKILLS

Applications do exist for promoting the development of programming skills. This task documents related work in the context of other R&D activities at a European and international level.

Visual Programming Platforms

This section is focused on presenting and comparing different platforms/programming environments that can be used for teaching programming skills to children. The following table is a summary of the different platforms that were considered.

Table 1 List of programming environments for building programming skills

	PLATFORM	TARGET GROUP	LANGUAGE(S)	FREE/PAID
Scratch	Web-based (but with offline version)	8-16	Visual computer programming language	Free
Snap!	Web-based (but with offline version)	12-20	Visual computer programming language	Free

Alice	Windows, Mac OS X, Linux	12-20	Visual computer programming language	Free
Tynker	Web-based, iOS	7+	Visual computer-programming language, JavaScript, Python	Paid
App Inventor	Web-based (but with offline version)	8+	Visual computer programming language for creating Android Apps	Free

Scratch

Scratch can be used as an introduction to programming, especially among younger students, but also to facilitate the transition to other programming environments, introducing and fostering interest in computer science (Scratch, n.a.) (Wolz, 2008). One of the benefits of the Scratch website is how it supports a worldwide network of users, hosting a community of programmers which allows users to share projects (Wolz, 2008) (Meerbaum-Salant, 2013). Scratch is available in more than 50 languages, including Bulgarian, Croatian, English, Greek, Italian, Portuguese, Slovenian and Turkish. It also has an offline editor that allows for the program to be downloaded to a computer and run without an internet connection. Prior to version 3.0, Scratch was using Flash but now is using HTML 5 and JavaScript.



Figure 5 Scratch user interface

Scratch is a block-based programming environment – these are visual programming languages that work as puzzle pieces, which allow for learners to assemble programs by snapping together the blocks and to receive visual feedback (Weintrop, 2015). Another benefit is that it allows for learners to familiarize themselves with the fundamentals of programming without having to worry about syntax (Ouahbi, 2015).

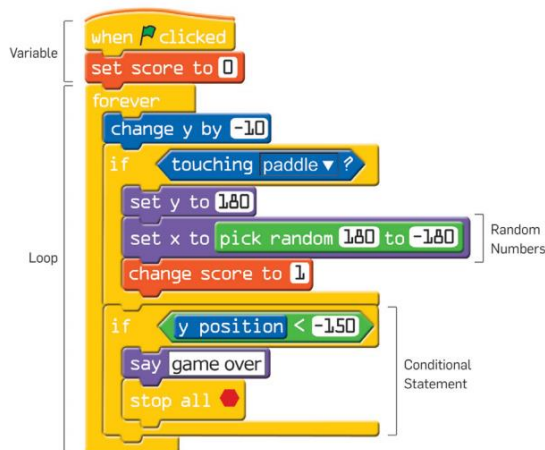


Figure 6 Building blocks of code in Scratch

According to a research study conducted by Meerbaum-Salant et al (2013), most students can achieve a reasonable level of Computer Science concepts through Scratch. However, difficulties arise when teaching specific topics that require a greater level of abstraction. This can, however, be solved if students are accompanied in the process by the teacher. Otherwise,

the study concluded that most students will only use it as a tool to create media and not as a way to learn programming as originally planned.

Snap!

Snap! is also a visual block-based programming language whose design is based on Scratch but adding some additional features (Snap!, n.a.). Snap!, similarly to Scratch, is web-based, but also has the possibility to be run offline through a browser. It is using HTML5 and JavaScript.

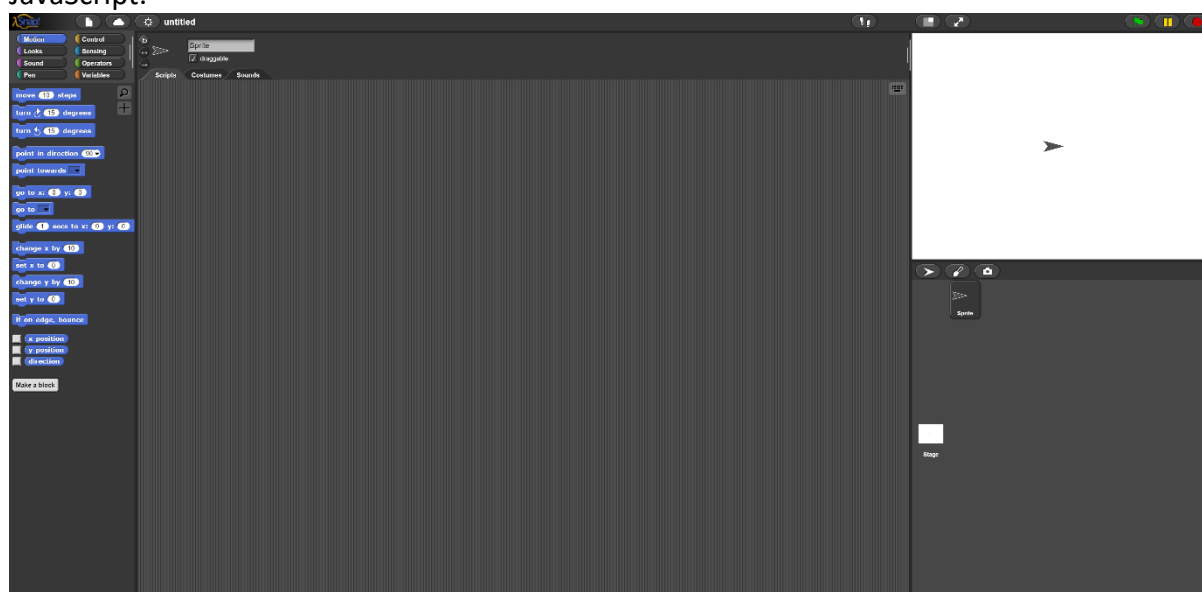


Figure 7 Snap! user interface

Apart from the features of Scratch, Snap! adds first class lists, first class procedures, first class sprites, first class costumes, first class sounds and first class continuations, thus making it more suitable for older audiences and as an introduction to computer science than Scratch. Snap! is available in over 40 languages, including Bulgarian, Croatian, English, Greek, Italian, Portuguese, Slovenian and Turkish.

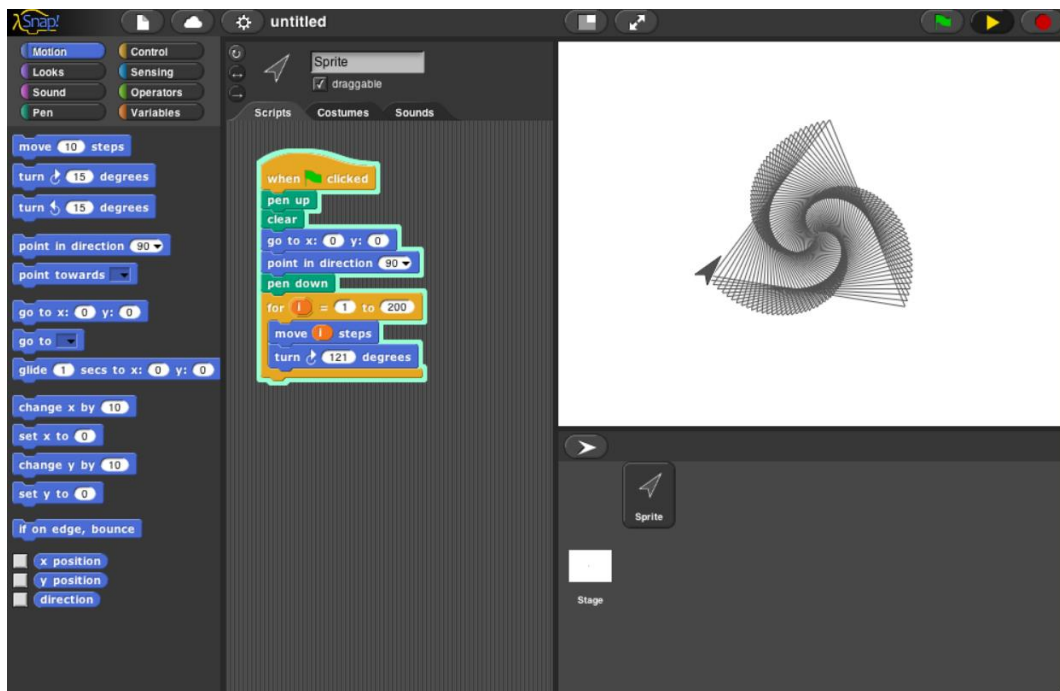


Figure 8 Building blocks of code in Snap!

A study was conducted by Weintrop et al (2015) with high school students using Snap! and Java. The students found the blocks-based approach to be easier than Java – thus, it is in accordance with the view that blocks-based programming (like Scratch and Snap!) is more accessible to novice programmers. According to the study findings, this was due to the fact that blocks are easier to read, due to the visual nature of the blocks that provide cues on how they can be used, they are easier to compose, and serve as memory aids.

Alice

Alice is a block-based programming environment that aims to motivate learners to program by involving their creativity, through the creation of animations, interactive narratives or simple 3D games (Alice, n.a.). As the other programming environments shown, it removes syntax errors by allowing learners to construct by dragging and dropping code elements (Kelleher, 2007).

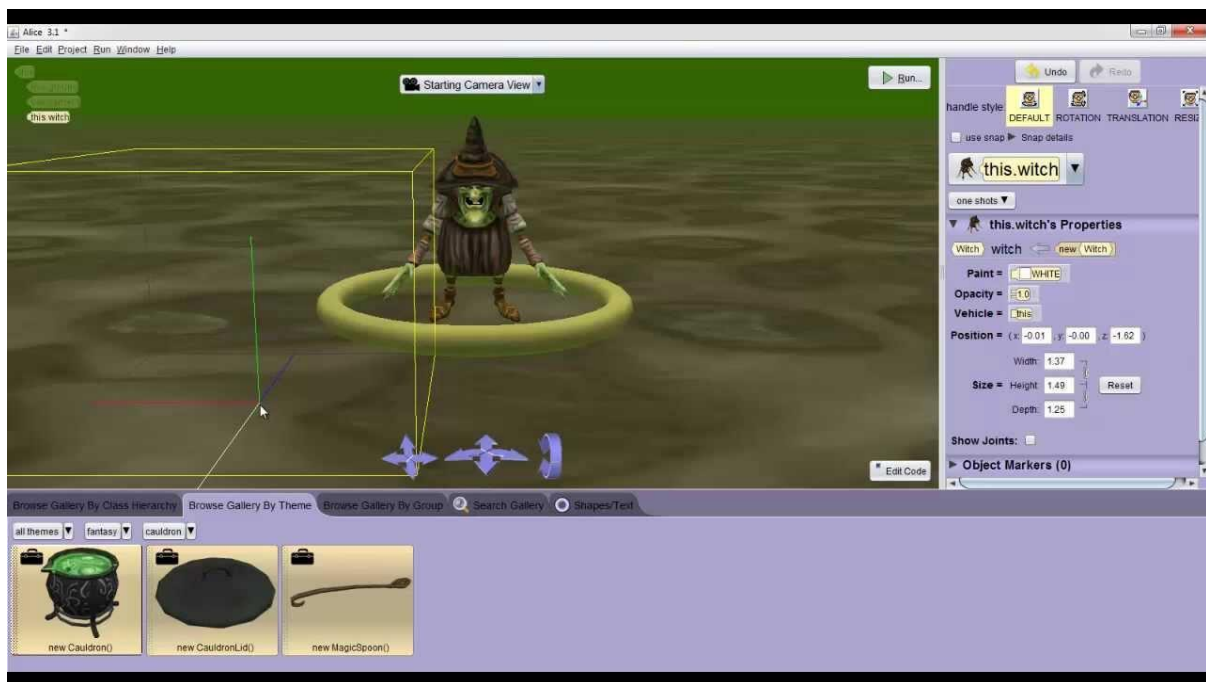


Figure 9 Adding objects in Alice

It allows for the creation of virtual worlds with a Virtual World Editor where learners can add 3D objects and add functions and methods to existing 3D objects. Once the Virtual World is created the learner can write code to develop the logic of the game/narrative/animation (Sykes, 2007).

Alice is a good programming language for learners with no previous experience, as it allows for them to see as they write their code how the animated programs run (Cooper, 2000).



Figure 10 Alice user interface

Moreover, a study that focused on Storytelling Alice (a programming environment based on Alice 2) which facilitates the creation of stories by students (with high-level animations, a storytelling gallery, and a story-based tutorial) found that although both Generic Alice and Storytelling Alice were equally successful at teaching programming concepts to girls, with Storytelling Alice, girls were more motivated and increased their time programming, although they found both equally entertaining. Motivation and time spent programming have a positive impact in programming performance – thus making programming more motivating to girl and maximizing the time they spent doing so may be a way to increase women’s participation in computer science.

Alice 3, the most recent instalment of the series, is available in 13 languages, including English, Portuguese, Greek, Slovenian, and Bulgarian, although each language might not be available at the same level of completion. Alice 2 is available in English, Spanish, Portuguese and German.

Tynker

Tynker is an educational programming platform based on a drag-and-drop visual programming language like Scratch. Unlike all the other programming environments presented, this one is a commercial product (Tynker, n.a.).

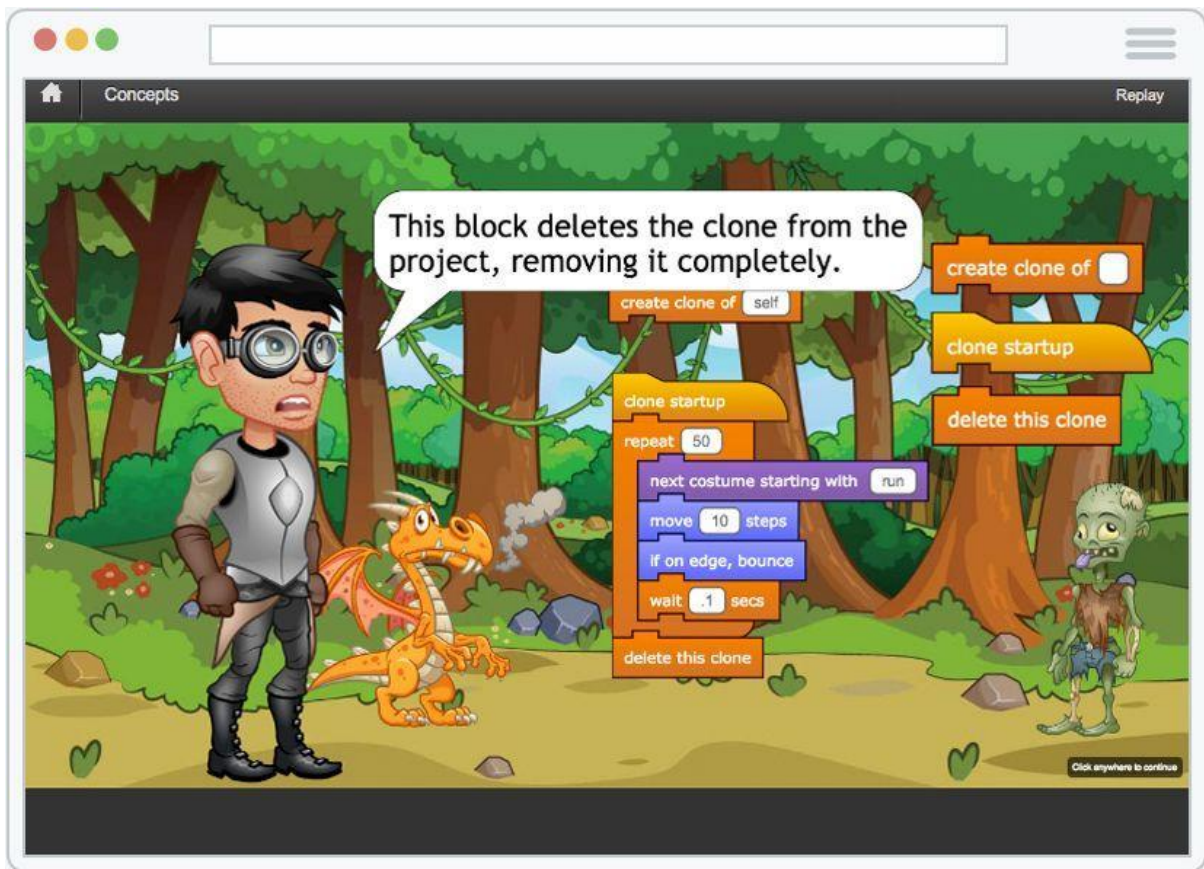


Figure 11 Tynker User Interface

One aspect it adds to Scratch consists in how it approaches coding as a game, in which the users are required to create a program in order to make the screen characters move, interact, and achieve different tasks. It presents problems the players need to solve, making it so that children begin to recognize patterns (Geist, 2016). Another addition to Scratch is the possibility to see equivalent of each code in JavaScript.

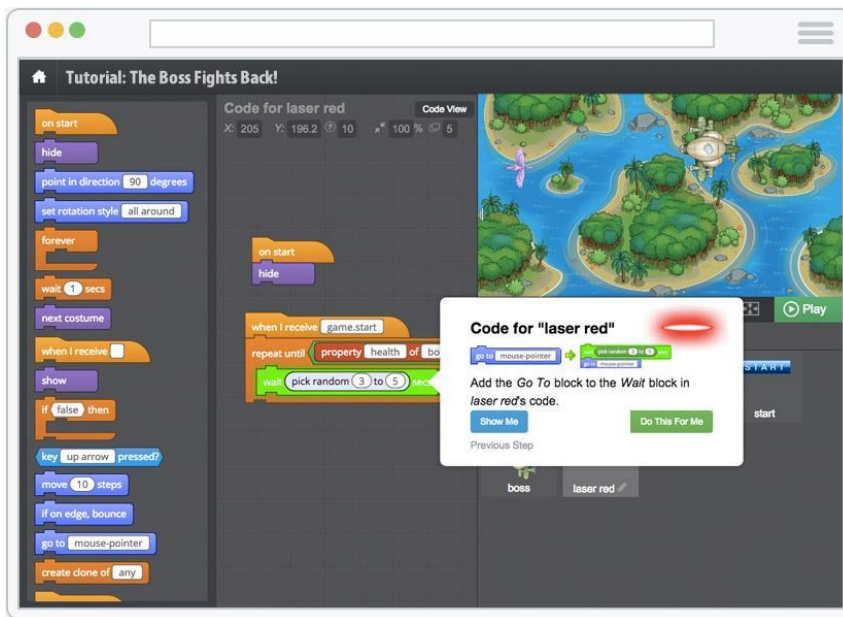


Figure 12 Building blocks of code in Tynker

Tynker also provides a built-in tutor to give step-by-step instructions, so that the learner can learn how to apply coding concepts. The learner is also able to visualize the blocks in JavaScript code, enabling them to understand the block in a text-based programming language (Tynker, n.a.).

Tynker provides over 23 Programming Courses, 11 iPad courses and 2000+ coding activities. It provides individual plans and family plans (for up to 4 members), that can be billed quarterly (20\$/month), yearly (10\$/month) or onetime payment (240\$). It is only available in English.

App Inventor

MIT App Inventor is a web application integrated development environment originally provided by Google, and now maintained by the Massachusetts Institute of Technology (MIT). MIT App Inventor is an intuitive, visual programming environment that allows everyone from children to seniors to build fully functional apps for smartphones and tablets. Those new to MIT App Inventor can have a simple first app up and running in less than 30 minutes. The blocks-based tool facilitates the creation of complex, high-impact apps in significantly less time than traditional programming environments. The MIT App Inventor project seeks to

democratize software development by empowering all people, especially young people, to move from technology consumption to technology creation.

Palette: Find your components and drag them to the Viewer to add them to your app.

Designer Button:
Click from any tab to go to the Designer tab.

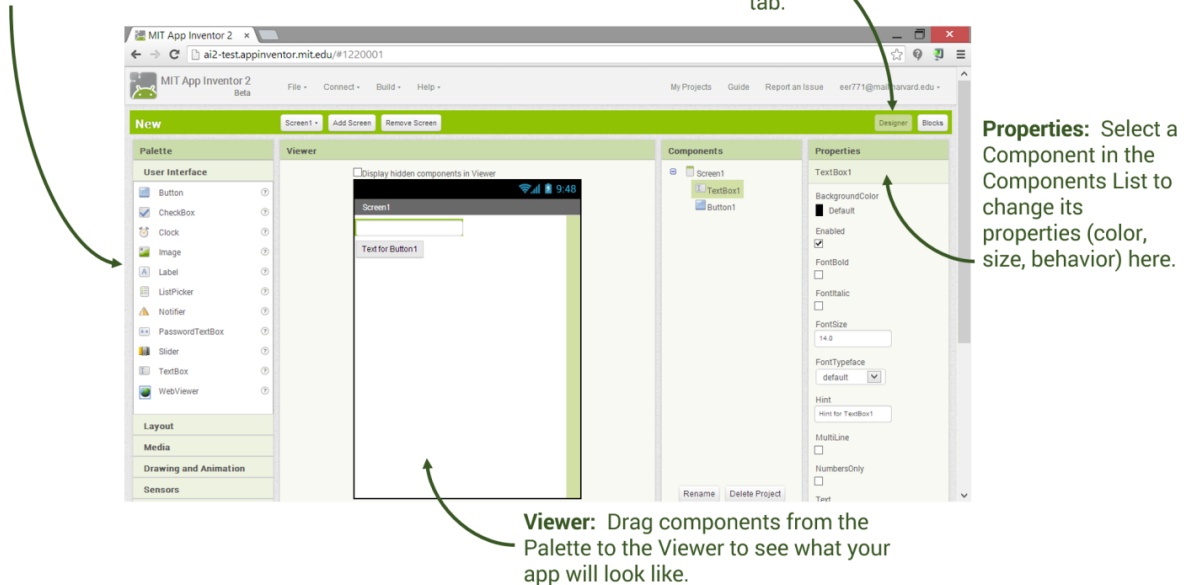


Figure 13 App Inventor user interface(1)

App Inventor and the other projects are based on and informed by constructionist learning theories, which emphasize that programming can be a vehicle for engaging powerful ideas through active learning. As such, it is part of an ongoing movement in computers and education that began with the work of Seymour Papert and the MIT Logo Group in the 1960s, and has also manifested itself with Mitchel Resnick's work on Lego Mindstorms and StarLogo.

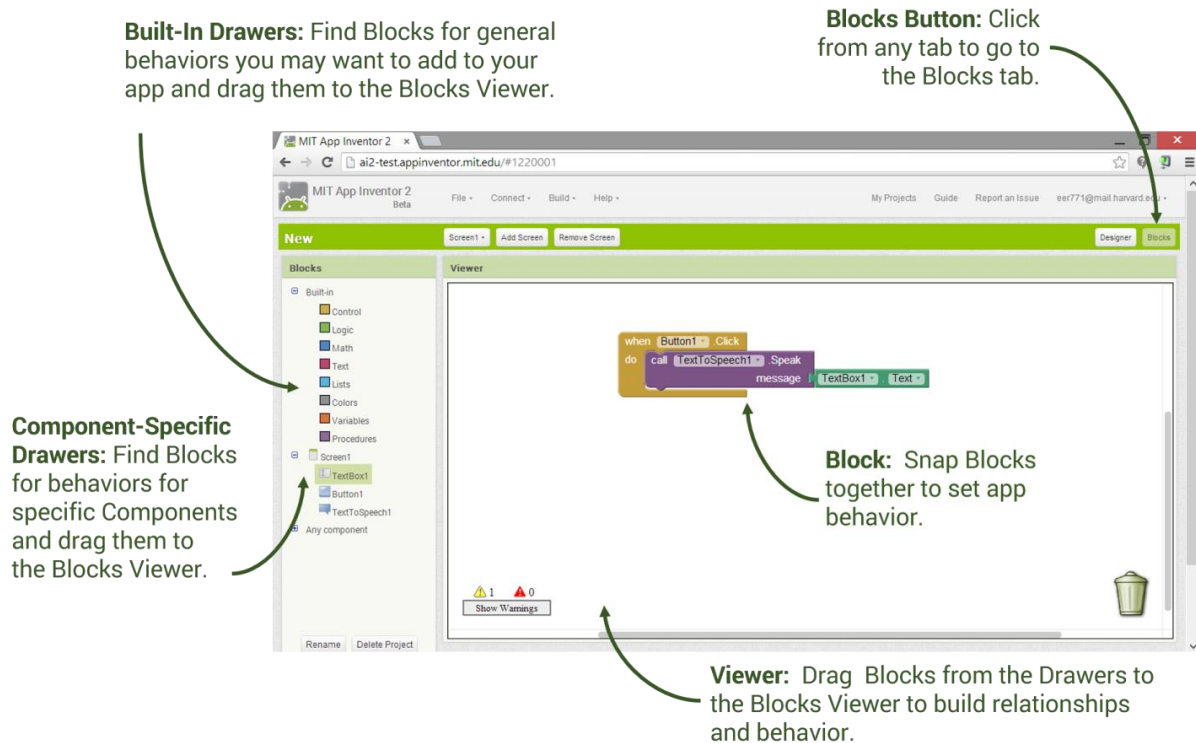


Figure 14 App Inventor user interface(2)

It uses a graphical user interface (GUI) very similar to the programming languages Scratch (programming language) and the StarLogo, which allows users to drag and drop visual objects to create an application that can run on android devices. App Inventor also supports the use of cloud data via an experimental Firebase (Firebase Realtime Database) component. It also supports many other components, and presents a special focus on AI. MIT is building tools into App Inventor that will enable even beginning students to create original AI applications that would have been advanced research a decade ago. This creates new opportunities for students to explore the possibilities of AI and empowers students as creators of the digital future.

Several scientific papers and presentation are using App Inventor as framework of study (Hsu, 2021a) (Zhu, 2021) (Yu, 2021) (Dunand, 2021) (Van Brummelen, 2021) (Hsu, 2021b).



Figure 15 Building blocks of code in App Inventor

No-code Platforms

No-code development platform (NCDPs) allows programmers and non-programmers to create application software through graphical user interfaces and configuration instead of traditional computer programming. No-code development platforms and low-code development platforms are closely related as both are specifically designed to expedite the application development process. These platforms have both increased in popularity as companies deal with the parallel trends of an increasingly mobile workforce and a limited supply of competent software developers. Offering a whole host of templates and pre-made solutions covering the most common user cases, those platforms can allow the users to produce a program that covers their needs extremely quickly, without possessing any coding skills and limited computer knowledge.

No-code development platforms are more specifically targeted towards the needs of small businesses wanting to deploy quickly a software solution to the issues they are facing. Contrary to the diverse solutions of low code development presented before, no-code platforms are more specifically geared towards productivity than education and their accessibility is based on a very commercial approach of things. All the no-code solutions require to be used online through proprietary interfaces and tools; none offer an offline or downloadable standalone applications. Very few of them present a free plan to use and when they do that plan presents very severe limitations (only one product can be made, only a few functionalities are available, the platform is only accessible for a certain amount of time and so forth).

In order to facilitate use of their platform and the interoperability of their products, many no-code solutions offer the possibility of adding many plugins developed by third party, either to increase quality of life, offer new functionalities or bridge or help integrate industry standards.

Some no-code platforms also offer the possibility to work collaboratively on the same product by offering built-in tools that facilitate communication and interaction between members of a same team.

Airtable

Airtable is a software platform that allows people to build the solutions they need to drive innovation and increase agility within their teams. This platform takes the approach of presenting the creation of a website or web application under the guise of a database interaction, putting the power of a flexible database into the hands of creators. At first glance, Airtable does indeed look a lot like a spreadsheet, but the tools offered are much more powerful. Building blocks enable teams to model the things they work on, define relationships between things, and create views explicitly tailored for their type of work. To avoid the need to form a single large table when there is related data in multiple tables, Airtable provides an option to link records of different tables. It allows linking existing tables of related records, creating a new linked table and also multiple links between existing tables

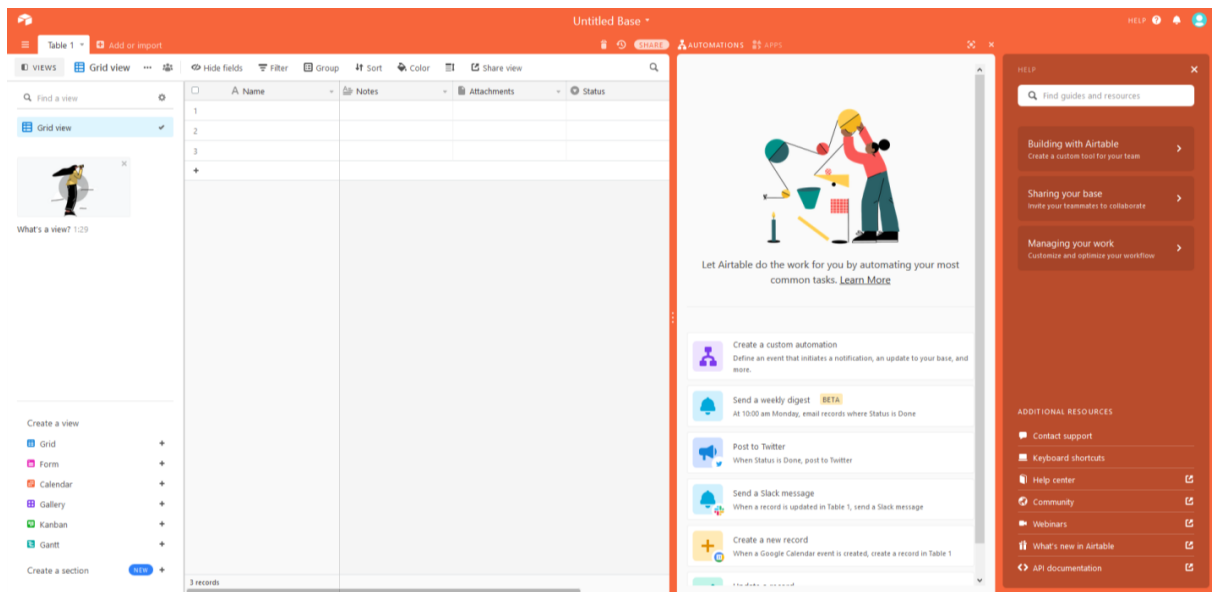
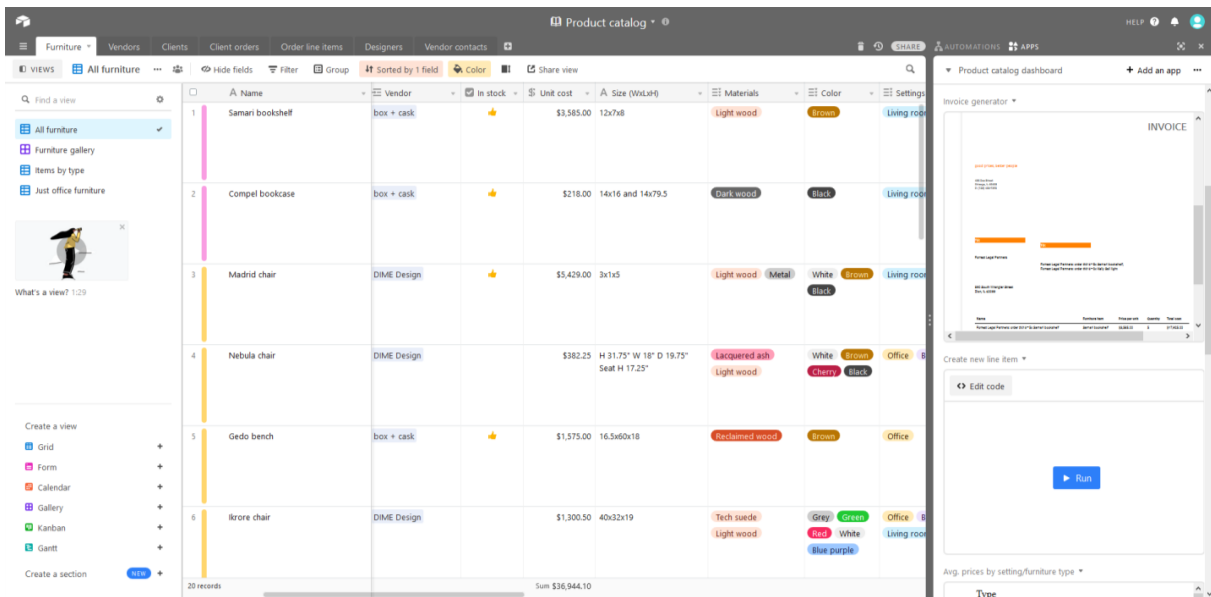


Figure 16 Airtable user interface

The platform increases agility by providing transparency and visibility to collaborators and managers by providing many tools related to team communication and management such as permission levels, calendars or kanban tools. The vast array of fully customizable templates available helps the users to solve a broad set of use cases without IT resources.

The platform is based on a fully graphical user interface, requiring filling boxes, ticking boxes, selecting elements in drop-down lists, to drag and drop objects and all other basic interactions common in a digital office environment that all users are already familiar with.

Airtable's API can be used to connect to other web services by which information can be exchanged between external web applications and Airtable.



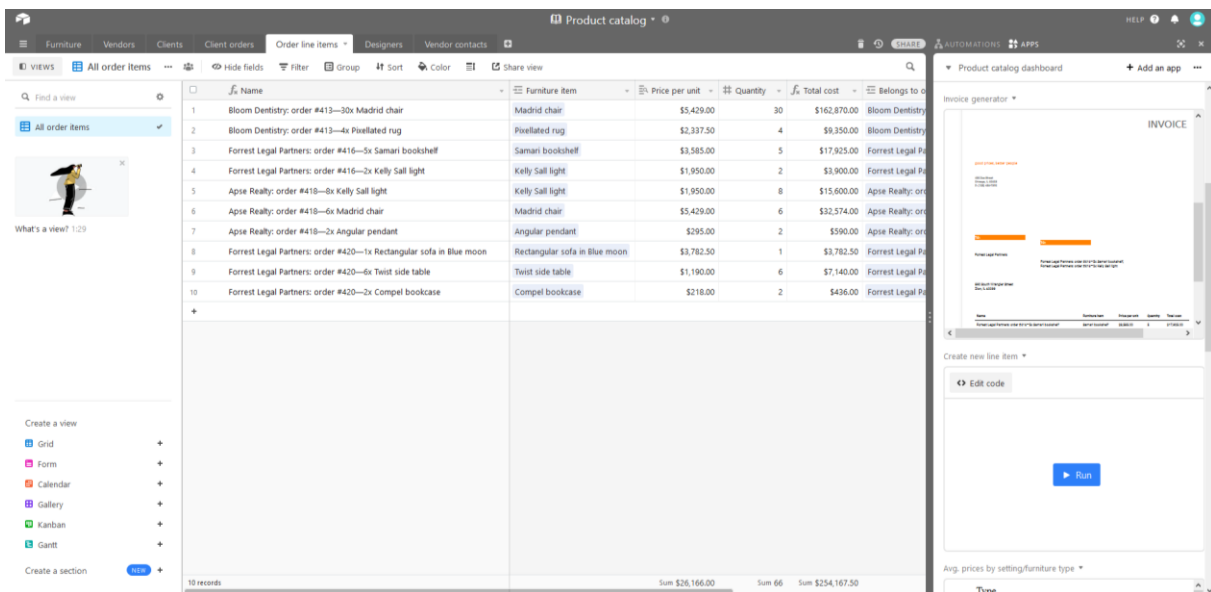
Name	Vendor	In stock	Unit cost	Size (WxHx)	Materials	Color	Settings
1 Samari bookshelf	box + cask	👍	\$3,585.00	12x7x8	Light wood	Brown	Living room
2 Compel bookcase	box + cask	👍	\$218.00	14x16 and 14x79.5	Dark wood	Black	Living room
3 Madrid chair	DIME Design	👍	\$5,429.00	3x1x5	Light wood	Metal	White, Green, Black
4 Nebula chair	DIME Design		\$382.25	H 31.75" W 18" D 19.75" Seat H 17.25"	Lacquered ash	White, Green, Cherry, Black	Office
5 Gedo bench	box + cask	👍	\$1,575.00	16.5x60x18	Reclaimed wood	Brown	Office
6 Irrore chair	DIME Design		\$1,300.50	40x32x19	Tech suede	Grey, Green, Red, White, Blue purple	Office

Figure 17 A database of objects in Air Table

The free plan does provide access to the platform but in a limited fashion with only limited amounts of records per database or size limits on the objects to be included in the data bases.

Pay-for plan weave those limitations and add additional support and functionalities.

A mobile version of the Airtable platform also exists on iOS and Android.



Name	Furniture item	Price per unit	Quantity	Total cost	Belongs to
1 Bloom Dentistry: order #413—30x Madrid chair	Madrid chair	\$5,429.00	30	\$162,870.00	Bloom Dentistry
2 Bloom Dentistry: order #413—4x Pixelated rug	Pixelated rug	\$2,337.50	4	\$9,350.00	Bloom Dentistry
3 Forrest Legal Partners: order #416—5x Samari bookshelf	Samari bookshelf	\$3,585.00	5	\$17,925.00	Forrest Legal Partners
4 Forrest Legal Partners: order #416—2x Kelly Sall light	Kelly Sall light	\$1,950.00	2	\$3,900.00	Forrest Legal Partners
5 Apise Realty: order #418—6x Kelly Sall light	Kelly Sall light	\$1,950.00	8	\$15,600.00	Apise Realty
6 Apise Realty: order #418—6x Madrid chair	Madrid chair	\$5,429.00	6	\$32,574.00	Apise Realty
7 Apise Realty: order #418—2x Angular pendant	Angular pendant	\$295.00	2	\$590.00	Apise Realty
8 Forrest Legal Partners: order #420—1x Rectangular sofa in Blue moon	Rectangular sofa in Blue moon	\$3,782.50	1	\$3,782.50	Forrest Legal Partners
9 Forrest Legal Partners: order #420—6x Twist side table	Twist side table	\$1,190.00	6	\$7,140.00	Forrest Legal Partners
10 Forrest Legal Partners: order #420—2x Compel bookcase	Compel bookcase	\$218.00	2	\$436.00	Forrest Legal Partners

Figure 18 A database of orders in Air Table

Appy Pie

Appy Pie is a straightforward app builder that targets Android, iOS and FireOs. It does allow its users to create mobile applications that can use monetization. It also offers its own market of example applications to build upon.

The interface of Appy Pie is very easy to use, mostly based on forms to fill and drag and drop options to select from.

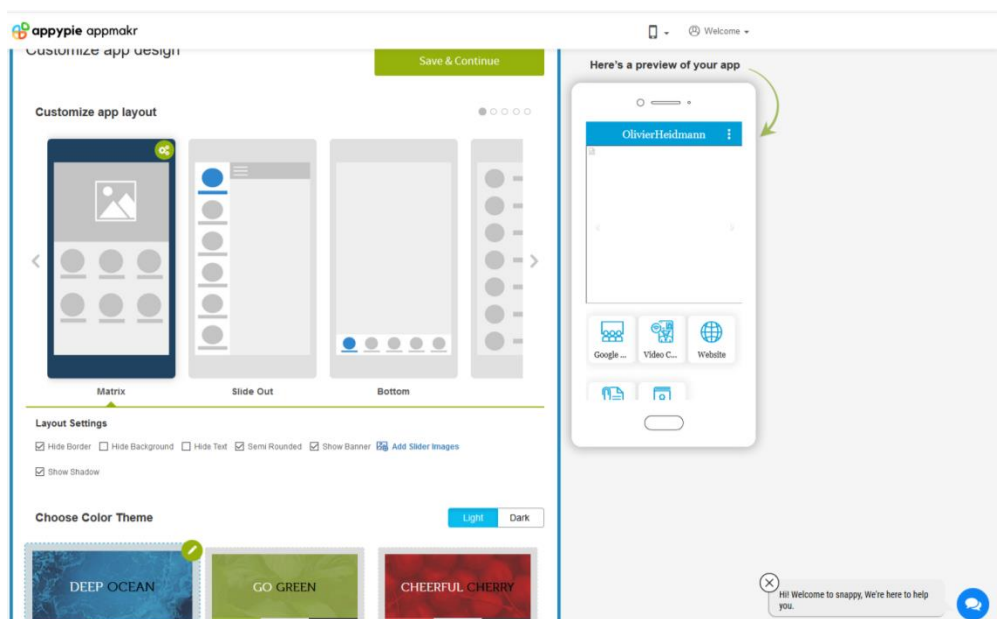


Figure 19 Appy Pie user interface

There are some limitations with some applications, such as the absence of plugins to link new social media platforms.

The free version offers only a demonstration of the platform possibilities, the basic plan starts at 16 dollars per app per month. The platform also offers The Student App Developer Program designed for an educational environment.

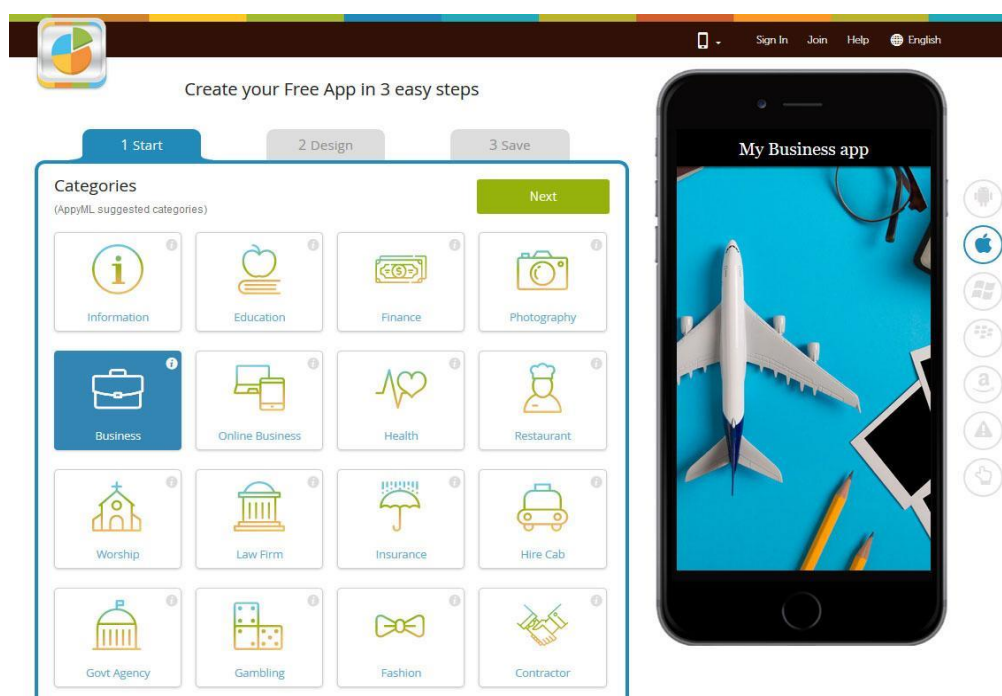


Figure 20 Appy Pie first steps to create an app

The Appy Pie platform also offers tools to build other types of product such as websites and chatbots.

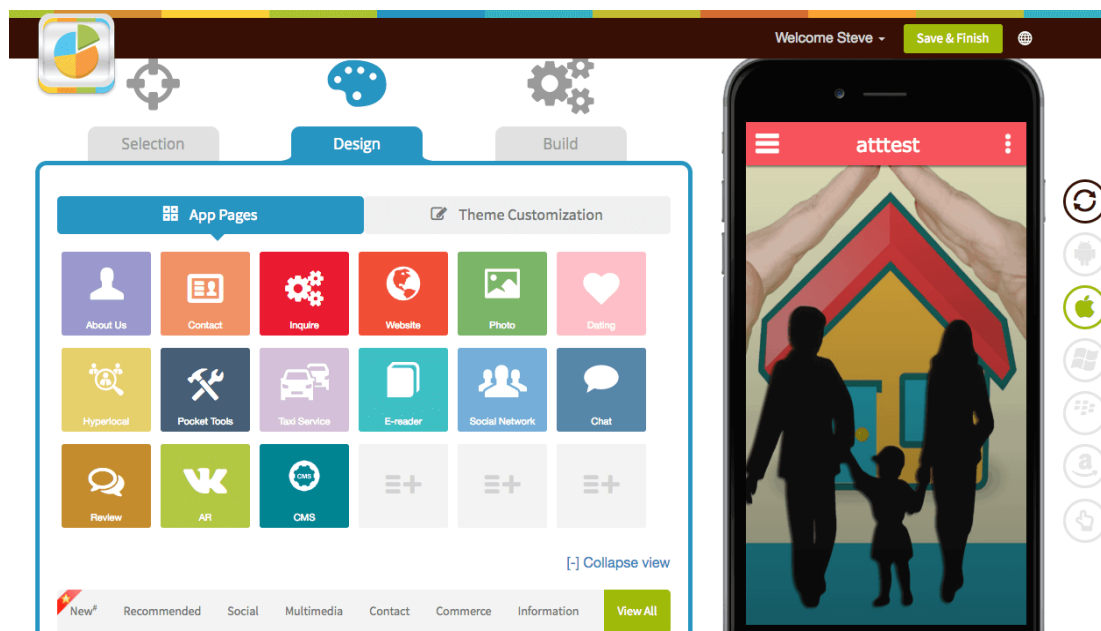


Figure 21 Creating an app in Appypie

Bubble

Bubble is a platform that allows users to design, develop and host interactive, multi-user apps for desktop and mobile web browsers. It can help create minimum viable products (MVPs) and full-fledged web apps with a mobile-friendly design. It is especially powerful when designing prototypes and MVP but might encounter some issues when scaling the application for a much larger use.

The interface is very intuitive with a drag-n-drop editor. Bubble users develop both the graphical interface and the user experience of their application by building out the logic and determining what happens when each element is interacted with. The platform contains a library of visual elements that can easily be inserted into an application for immediate use. There is also a variety of free and paid-for templates created by Bubble designers that are available to the users and more than a 1000 plugins to connect the application to external services.

Bubble apps come with a ready-made user management system that makes setting up user accounts and log-ins (including with common platforms such as Facebook, Google or LinkedIn) an easy task. The platform also proposes tools to translate an app's text and automatically display the right language, currency, and formats for website messages without additional work. Bubble supports 80+ international languages and it is possible to add more.

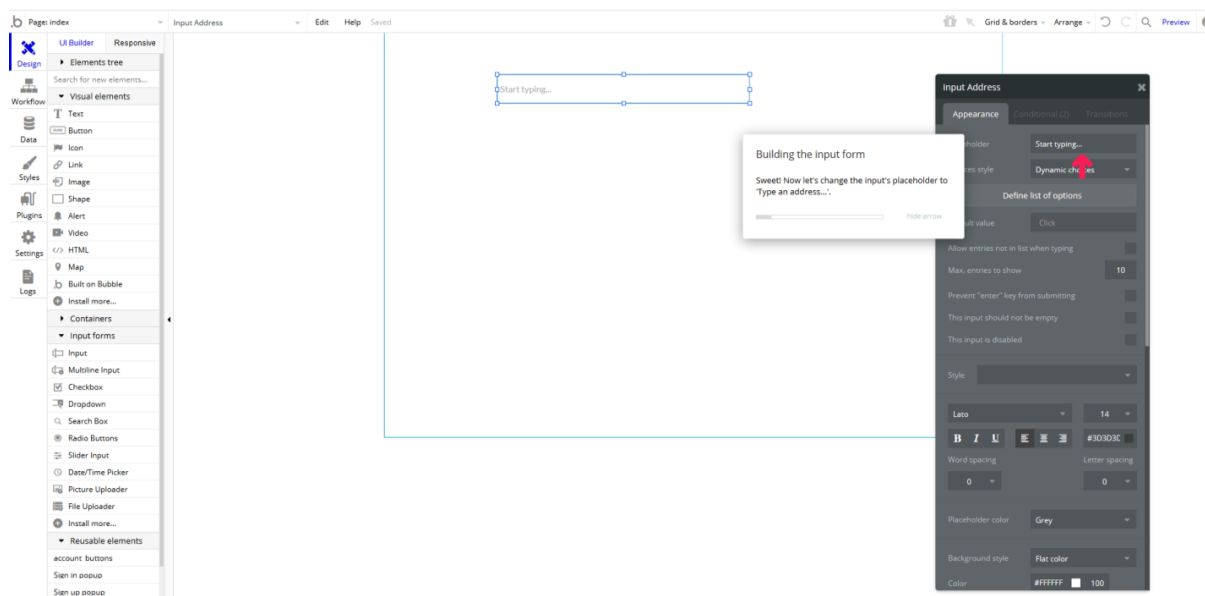


Figure 22 Bubble user interface

Up to 40 teammates can be working together on a given application, with customizable permissions and access rights. Collaborators can see what each other are modifying live and provide feedback by marking up the app with notes, tasks, and comments for the entire team to view.

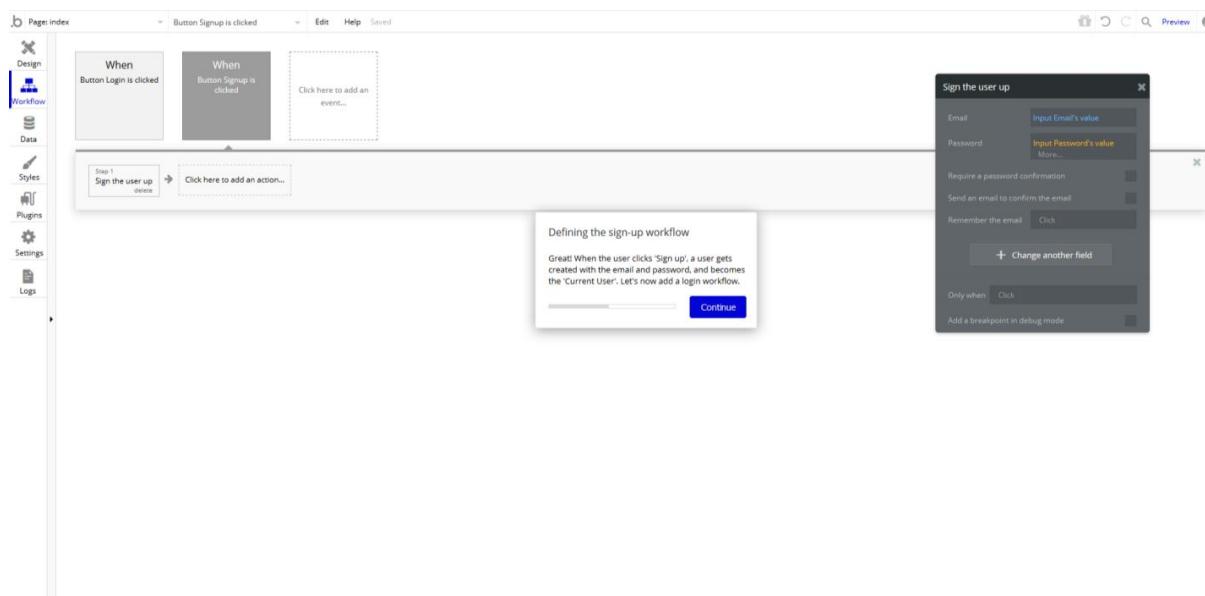


Figure 23 Designing the workflow of the webapp in Bubble

The free plan offers only limited functionalities compared to the pay for plans and a separate plan is needed to build each new app.

The interface of Bubble is very rich and comprehensive but it might also be seen as complex to master.

Gamified Programming Platforms

Since their apparition on the consumer market less than 50 years ago, computer games have slowly but steadily become a crucial element in our social and cultural environment (Oblinger, 2004) (Oblinger, 2006). The purpose of playing video games already extends beyond their traditional entertainment value and they have been used in many different contexts, like the medical field (e.g. as a tool for repairing social link) or the military sphere (e.g. as a recruiting tool). One of the most common and ever increasing use of video games happens in the field of education, as a learning tool providing significant benefits to all stakeholders in the classroom context and outside of it. It promotes interaction from students with complex, risk-free, skill-demanding practices, it increases motivation and engagement, while at the same time strengthening psychomotor skills, enhancing knowledge retention and decision-making skills, and providing opportunities for repeated practice and immediate feedback (Tashiro, 2007).

The benefits and usage of playing in the framework of learning predates the invention of video games but the ubiquitous presence of computers in all aspects of our current society has bolstered the apparition and development of a dedicated branch of video games dedicated to be used in an educational setting, called serious games. Although the exact borders between commercial computer games and serious games is sometimes fuzzy and the video game landscape of genre and definitions is always ever changing, some formal descriptions of what serious games are exactly have been put forward. One stipulates that serious games can be defined as games “...that do not have entertainment, enjoyment or fun as their primary purpose” (Chen, 2006). Another definition can be that serious games are “a mental contest, played with a computer in accordance with specific rules, that uses entertainment to further

government or corporate training, education, health, public policy, and strategic communication objectives” (Zyda, 2005).

Building on the core principles the design of commercial video games has laid down through decades of iterations, serious games aim at extracting components that make gaming appealing, and combine these with the desired information and knowledge to be transmitted to the user, creating an interactive source for learning that, in turn, motivate each user to extend their own knowledge and deepen their study in a challenging, fun and instant approach (Prensky, 2003).

As a study by Freitas et al. (2007) concluded, serious games are an advantageous tool in a teacher’s toolbox in the sense that they allow for the creation of content for the purpose of adapted, specific and tailored learning without taking any of the fun out of it.

As is it a vast subject that can be tackled in many different fashions, there exist a wide variety of approaches for teaching programming to novice coders. It can be done for example through programming exercises based on visual programming languages. Such languages are especially user friendly as they are both easy to operate (on a drag-and-drop basis) and present content in a very intuitive fashion. Furthermore, object oriented programming is by essence much more easy to introduce and teach in such an environment. Scratch (Resnick, 2009), Snap! (Weintrop, 2015), or Alice 2/3 (Alice, n.a.), among others are famous and successful visual programming languages as we’ve seen before.

Computer games can also be used to teach coding, either by encouraging students to develop and create their very own video games or by allowing them to play serious games whose learning outcomes encompass learning outcomes related to programming.

Students can also create their own basic games through those block-based environments. These drag and drop environments are useful, as they avoid errors regarding syntax and logic (Ouahbi, 2015). In a study that compared using the Scratch (visual based programming) and Pascal (command line or GUI based programming) coding environment with novice

programmers, they found that learners were more motivated to continue their studies in computer sciences when using Scratch. The study also found that the creation of games and stories makes learners more creative and autonomous while learning programming.

The idea behind using serious games to teach programming comes from the fact that these, by being more engaging and motivational, allow for students to learn computational thinking and programming skills in entertaining and familiar environments, before transferring those skills to learning a programming language.

According to Kazimoglu et al. (2012), serious games for learning programming and computational thinking should not only be created for fun, and should also consider a curriculum and skills, making a difference between different programming constructs and encouraging good programming practices (by ways of gamification, like achieving a high score).

Serious games can also be useful for programming by turning unpleasant operations into interesting experiences — Shabanah et al. (2010) have created an Algorithm Game Designer aimed at facilitating the creation of algorithm games, in order to improve engagement in algorithm visualization systems. Grivokostopoulou et al. (2016) have also developed a game for teaching AI algorithms, based on the Pacman game, so that through algorithm visualizations and animations students could be helped in their learning process, by demonstrating the way an algorithm operates, its functions and how to make proper decisions based on specific parameters.

Following the trends created by commercial video games, serious games have traditionally been deployed in an offline fashion. Technical issues and challenges specific to the world of education have also contributed keeping most of serious games offline. But with the developing widespread adoption of faster internet connection speeds and rising availability of computers in everyone's homes, online game-based platforms like CodeCombat (CodeCombat, n.a.), LightBot (LightBot, n.a.) and Scratch (Scratch, n.a.) have begun to appear. Combéfis et al. (2016) reviewed the main types of online platforms used for teaching

programming skills, and have concluded that the following factors can make a serious game more motivating and successful: 1) a feedback and assessment process; 2) aesthetics; 3) collaboration and multiplayer aspects; 4) guidance through the game; 5) no negative consequences; 6) music; 7) a medium-level of challenge to keep the interest of the players. The question of collaborative and competitive games for programming was also highlighted by Miljanovic et al. (2018).

Regarding the learning outcomes of Serious Games for Learning Programming, according to an extensive review of 49 serious programming games by Miljanovic et al., most serious games for programming focus on problem solving and fundamental programming concepts, with few games focusing on data structures, development methods and software design.

There are also games that, while they may not teach programming skills, are capable of teaching in an indirect but effective way key computational thinking skills, like abstraction (eg. Tetris), decomposition (eg. Sudoku), algorithm thinking (eg. Puzzle games), evaluation (eg. Memory Games), and generalization (eg. Pattern Games) (Franković, 2018).

The following table is a summary of the different games that were considered.

Table 2 List of some serious games for learning programming skills

GAME	PLATFORM	TARGET GROUP	LANGUAGE(S)	LEARNING UNITS	TYPE OF GAME
Code Combat	Web-based	9+	JavaScript, Python	syntax, methods, parameters, strings, loops and variables, If/else, Boolean logic, relational operators, functions, object properties, event handling, input handling, Arithmetic, counters, while-loops, break, continue, arrays, string comparison, finding min/max, Object literals, remote method, invocation, for-loops, complex functions, drawing, modulo	Commercial Game (Freemium)
Human Resource Machine	Windows, Mac OS X, Linux, iOS, Android	9+	Visual Programming Language-based	Input & output and conditionals & iteratives algorithm comparison	Commercial Game (Paid) Puzzle Game
Lightbot	iOS, Android, Windows, Mac OS X	9+	Visual Programming Language-based	Conditionals and Iteratives and Recursion Debugging Strategies	Commercial Game Paid Puzzle Game
May's Journey	Windows	12-18	Custom Programming Language (inspired from an	Basic instructions and sequence logic, loops, variables, if statements, comparators and	Research Game Puzzle Game

			Object-Oriented Language like Java)	Boolean logic, operations on integers, operations on strings	
No Bug's Snack Bar	Web-based	18+	Visual Programming Language	Variable Manipulation, Sequence of Actions, Conditional & Iteratives, Debugging Strategies	Research Game
Robot ON!	Windows	18+	C++	Syntax & Semantics, Variable & Primitive Data Types, Expressions & Assignments, Conditionals & Iteratives, Program Comprehension	Free/Research Game
Educational 'Pacman' Game	Windows	18+	Game for learning AI search Algorithm	Algorithms (Basic textual description of Algorithms and corresponding graphical flowchart along with Pseudocode)	Research Game
CMX	Windows	18+	C	Theory on arrays, outputs of variables after an array's program's execution, syntax of array programs, entry of variable values in an array, calculation of the array's sum, calculation of the maximum value of the array,	Research Game MMORPG

Code Combat

Code Combat is a web-based adventure game. It has plans for both individual students and classes, providing also resources for teachers to use during classes. It has more than 100 free-to-play and subscriber-only levels. Premium plans are available for \$3.99/month or a one-time payment of \$39.99. It is available in over 50 languages, including Bulgarian, Croatian, English, Greek, Italian, Portuguese, Slovenian and Turkish.

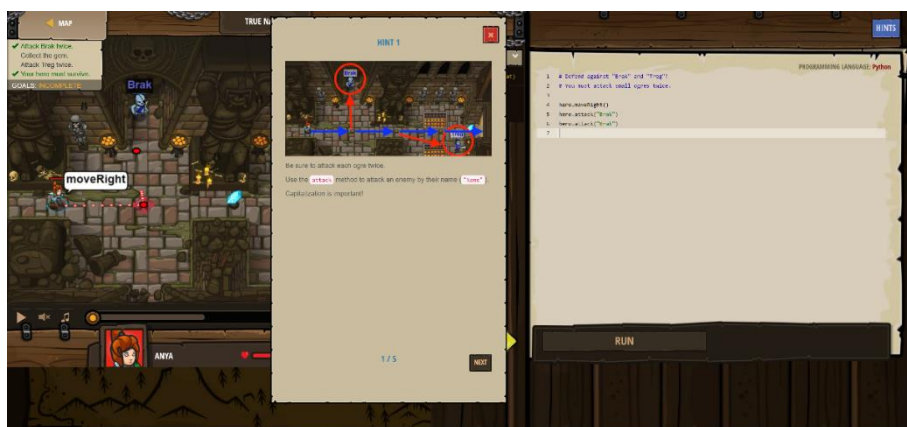


Figure 24 Code Combat Gameplay

Learners are required to write their own code (either in JavaScript or Python) in order to make the characters move, interact, and achieve different tasks. It is not block-based, requiring learners to write their own code. It provides hints along the way and introduces concepts

gradually. The game is divided in 5 different worlds, introducing different concepts as the player progresses through the game: 1) Kithgard Dungeon – syntax, methods, parameters, strings, loops and variables; 2) Backwoods Forest – If/else, Boolean logic, relational operators, functions, object properties, event handling, input handling; 3) Sarven Desert – Arithmetic, counters, while-loops, break, continue, arrays, string comparison, finding min/max; 4) Cloudrip Mountain – Object literals, remote method, invocation, for-loops, complex functions, drawing, modulo; 5) Kelvintaph Glacier – Advanced Techniques.

According to Miljanovic et al. (2018), the game's educational content comprises Algorithms & Design (Algorithm Comparison, Problem Solving), as well as Fundamental Programming Concepts (such as Syntax & Semantics, Variables & Primitive Data Types; Expressions & Assignments, Input & Output, Conditionals & Iteratives, Functions & Parameters, and Recursion), Fundamental Data Structures (such as Arrays, Heterogeneous Aggregates, and Abstract Data Types).

Human Resource Machine

Human Resource Machine is a puzzle game based on a visual programming language. In this game, the player has to automate a task by programming an office worker, being promoted to different levels as they progress through different puzzles (Johnson, 2016). It is a commercial game, being available on Steam for 8.99€.

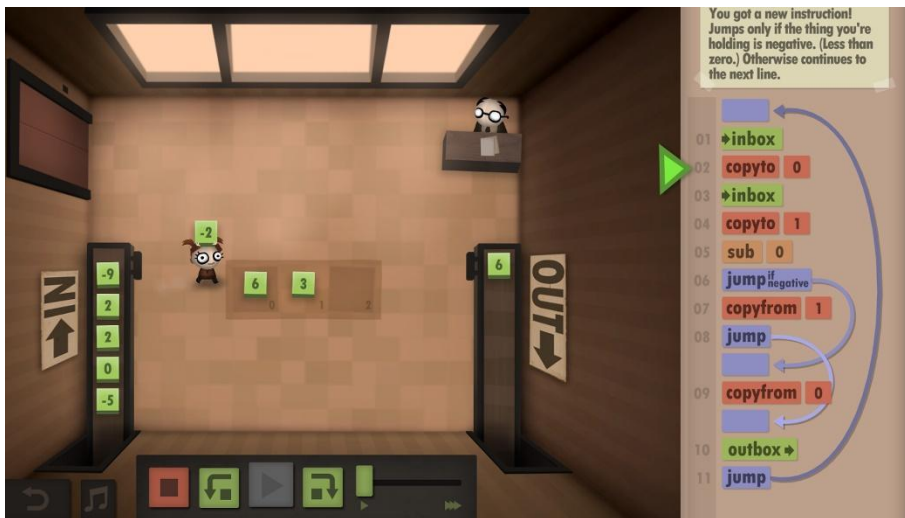


Figure 25 Human Resource Machine Gameplay

The player must create a sequence of moves, by dragging and dropping instructions, with new commands being gradually introduced to accomplish more complex operations. The first commands are -> inbox and outbox.

Through a visual programming language based in blocks, it teaches fundamental programming concepts, like input & output and conditionals & iteratives, as well as teaching algorithm comparison (Miljanovic, 2018). It is mostly useful for learning how to write assembly-code.

LightBot

LightBot is a puzzle game, with the mechanics being similar to that of Human Resource Machine, requiring programming logical in order to progress through the levels. The player must guide a robot in order to light up tiles and solve levels, featuring 40 levels and 20 challenge stars. The commands used in Lightbot appear as icons.

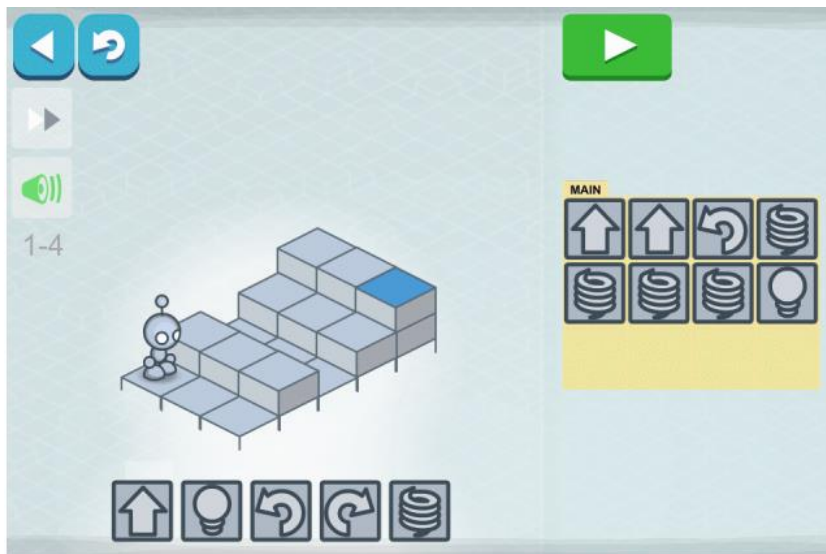


Figure 26 Lightbot Interface

Although in Lightbot, the players do not learn any programming language, “they develop an understanding of sequencing and implementation of algorithms”(p.6), focusing instead on acquiring problem solving skills (Miljanovic, 2018). Through the game, students learn different fundamental programming concepts, such as Conditionals and Iteratives and Recursion, as well as debugging strategies. As the space for the tiles is limited, the learner must also consider code size.

A study conducted by Mathrani et al. (2016) using Lightbot with students of a non-university education provider found that students enjoyed playing the games and that this was an effective way of learning programming constructs like functions, procedures, conditionals and recursions. Most participating students agreed that the approach was an effective way for them to understand concepts that would otherwise be more difficult.

May's Journey

May's Journey is the only game from this list that is directly targeted toward middle school girls. It is a 3D puzzle game in which the players solve a maze through the means of the game's custom programming language, which is inspired by Java (Jemmali, 2016).

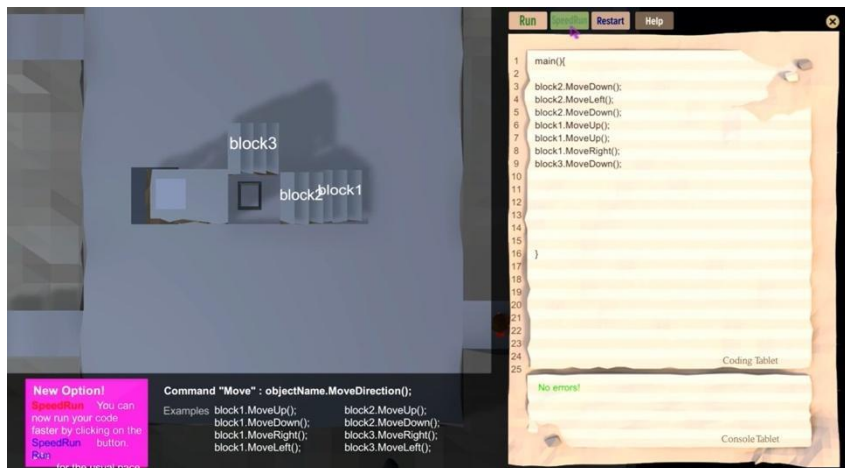


Figure 27 May's Journey Gameplay

The game was designed with middle school girls interests in mind, hoping to attract them to Computer Science by teaching the basics of programming. The developer employs a pseudo-code in order to facilitate the transition between visual programming (drag and drop programs like Scratch) and real programming languages. The teaching content is: basic instructions and sequence logic, loops, variables, if statements, comparators and Boolean logic, operations on integers, operations on strings.

There are two phases in the game, an exploration phase with the mechanics of a typical game and a coding phase. In the coding phase, the interface is split in order for the player to type the code, but also get visual feedback of the code. Hints for the code are also provided during the exploration phase. In the story, the hero is a girl who lives in a world that is falling apart and is separated from her friend, needing to fix the game world and solve mysteries. Each part of the mystery is revealed gradually, motivating the players to play more. As pre-teens are the main target group, they made the main character look like a middle school girl, so that the player can project themselves unto the girl. The design also had in mind girls preferences towards design (high lightness, warm colour scheme and less aggressive illustration. The test group seemed also to appreciate the storyline (Jemmali, 2016).

No Bug's Snack Bar

No Bug's Snack Bar is a research serious game with a drag and drop block-based approach to make the game independent from learning the syntax of the computer programming language, focusing on problem solving (Vahldick, 2017). With a focus on Problem Solving, its learning outcomes are: Variable Manipulation, Sequence of Actions, Conditional & Iterativess, Debugging Strategies.

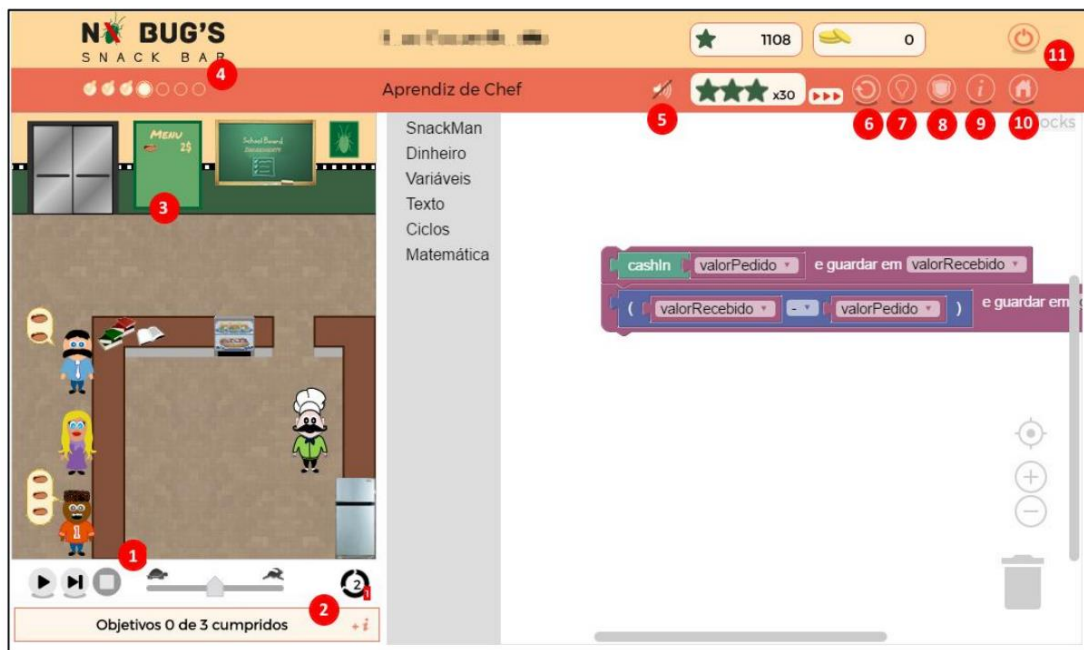


Figure 28 No Bug's Snack Bar Gameplay

In the game, the main character has to work at a Snack Bar, passing through different missions using code. One innovation included in this game is the fact that there's a tool incorporated for teachers to monitor how students are progressing through the game. There's also a gamification approach, with the student gaining points as they progress through the games and find ways to better their solutions to the problems presented in order to maximize the points gained.

The study afterwards found that the biggest fault within the game was the absence of a support system, with hints — therefore, the developer incorporated an assistant with hints and an administrative system for the teacher to see who are the learners that need help and

to send them personalized hints. The presence of a teacher was, therefore, seen as fundamental.

Robot ON!

Robot ON! is also a research puzzle-type game aimed at undergraduate students who are learning C++ (Miljanovic, 2016). In this game, the player is a scientist that has to activate a 'Mech Suit', doing so through a series of tasks. Once the player finishes a level, he activates a new robot system. It also has a framework in which instructors can create their own levels using other programming languages. The player has different tools, which are colour coded, with the colour on the code corresponding to the different tools. The player is introduced to different tools one at a time, accompanied by tutorials.



Figure 29 Robot ON! Gameplay

Instead of focusing on writing code, this game focus on programming comprehension, in order to teach debugging skills and understand code written by others. As the game progresses, the player is given the following tools: 1) activator tool (to learn control flow); 2) commenter and

un-commenter tools (to learn code behaviour); 3) namer tool (to learn variable purpose); and 4) checker tool (to learn data flow).

Educational Pacman Game

Educational Pacman Game is a research serious game designed to teach search algorithms, allowing for students to see how different search algorithms behave and a graphical annotated depiction of them (Grivokostopoulou, 2016).

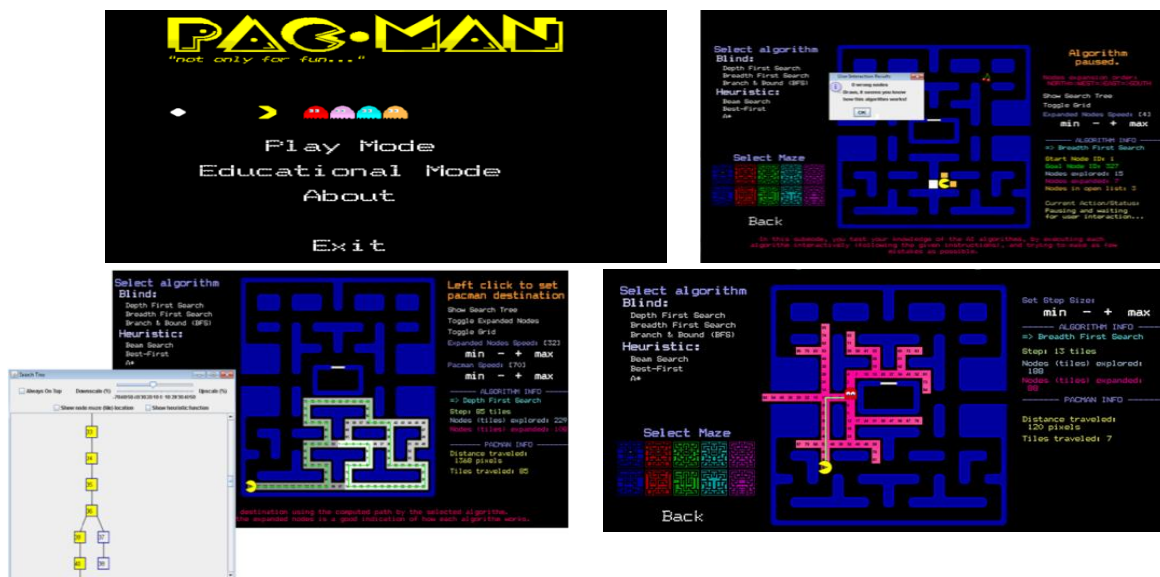


Figure 30 Educational Pacman Game

The game has two modes:

- 1) Educational Mode: the student can read a textual description and the graphical flowchart and pseudocode of an algorithm of his choice. The game also allows for the student to learn the algorithm via visualizations on the different Pacman Mazes.
- 2) Playing Mode: the student has to solve maze levels within different conditions and with a time constraint. These levels are made so that the student has to apply a specific search algorithm to move the Pacman in the maze — the student is asked to, from a random position, reach to a cheery or power-up by moving the character on the specific algorithm.

CMX

CMX is a Massive Multiplayer Online Role Playing Game (MMORPG) for learning programming (Malliarakis, 2014). In the game there are two teams — crackers and hackers — who compete against each other to find the passwords on a global toxic waste factory. They are assisted by Senseis that assist them in learning the programming language C.

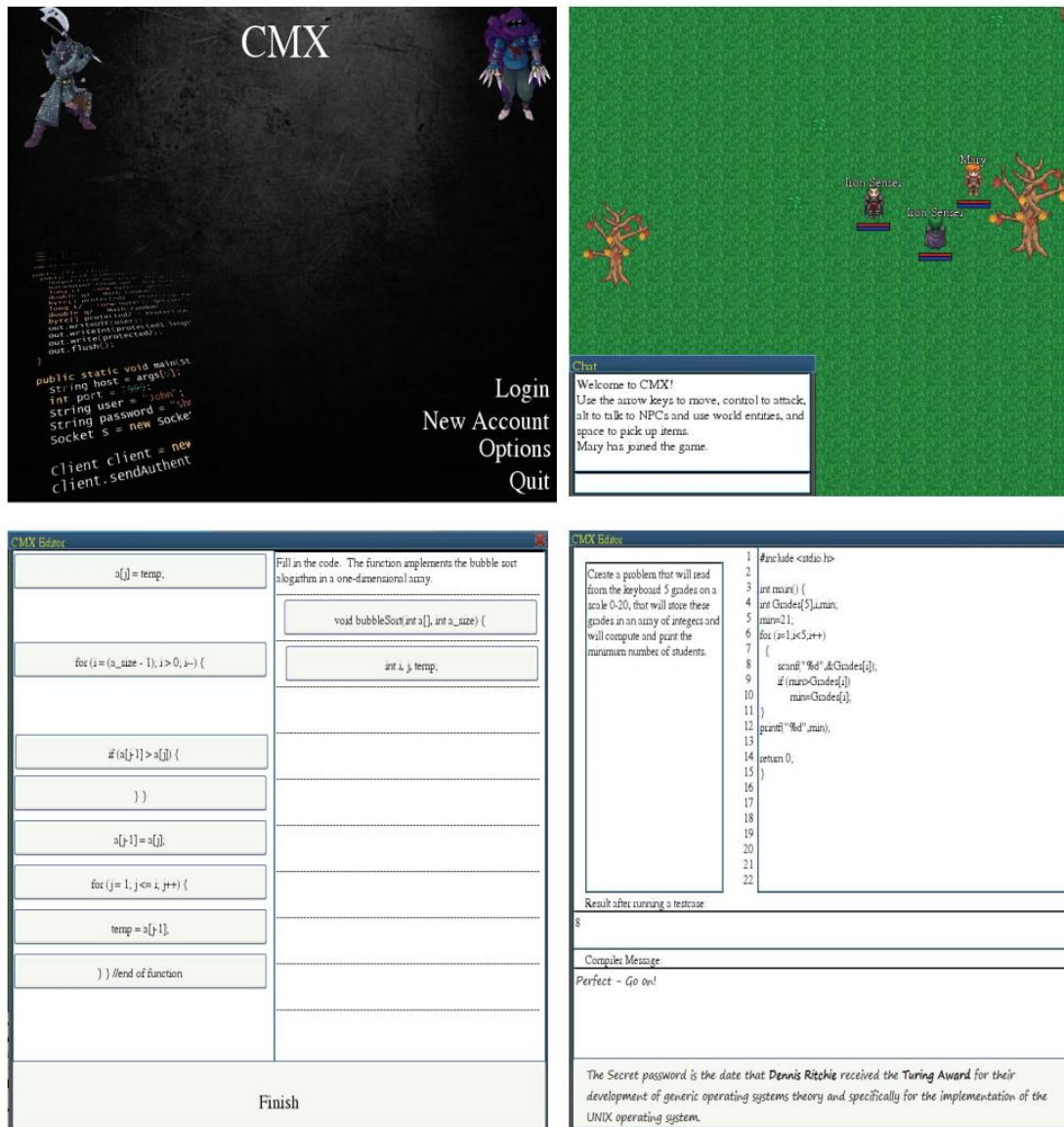


Figure 31 CMX Game Environment

The game includes three levels of Senseis, with each one holding one password and unlocking another Sensei

The students seemed to increase their comprehension level of C — they could not only answer theory questions, but also lay executable programs by dragging and dropping tools, as well as writing programs in C.

Low-Code Game Editors

By low-code game engines we are characterizing a set of applications that are primarily targeted at the development of games but can be used for other purposes, while requiring very little knowledge of programming or even allowing the user to develop those skills while creating games and other applications.

GameMaker Studio 2

GameMaker Studio 2 is a tool designed to make new and innovative games as well as prototype ideas in the fastest way possible across multiple target platforms. It is intended primarily as a tool for making 2D games - although 3D games are perfectly doable - and comes with a number of tools and editors to help realize ideas, with the final project being ported across multiple platforms from the same initial base resources.

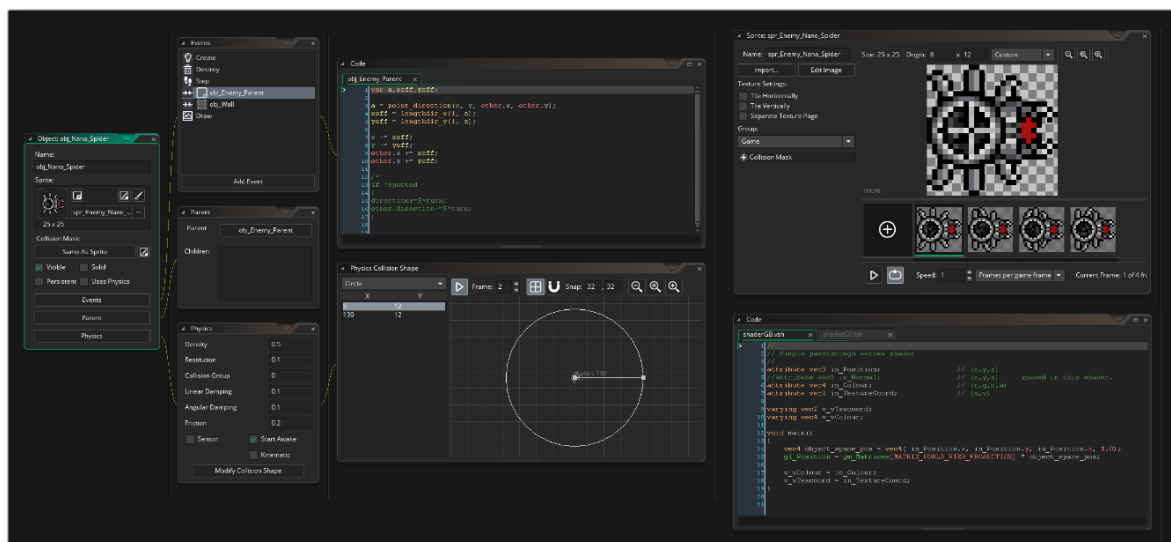


Figure 32 The GameMaker Studio 2 interface

For those that are new to the world of programming or who have never used any game creation tool before, GameMaker Studio 2 offers an intuitive and easy to use Drag and Drop interface of action icons that allows to start creating games very quickly using visual scripting.

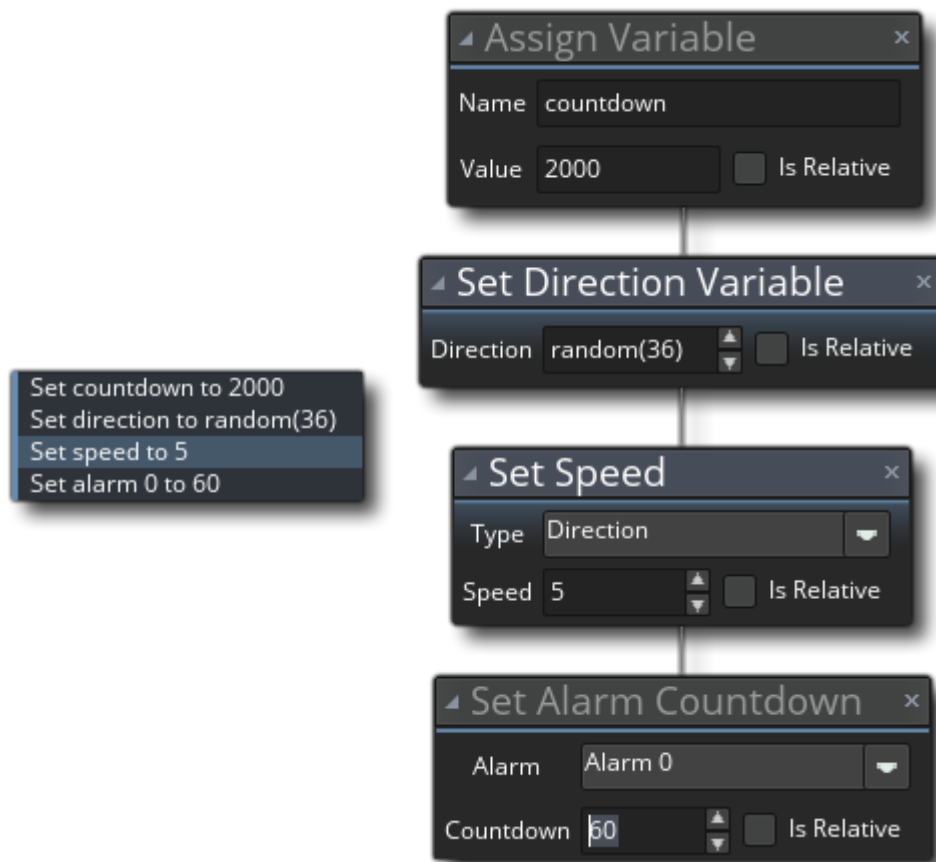


Figure 33 Blocks of code in Game Maker Studio 2

RPG MAKER

RPG Maker is a program that allows users to create their own role-playing video games. Most versions include a tile set based map editor, a simple scripting language for scripting events, and a battle editor. All versions include initial premade tilesets, characters, and events which can be used in creating new games. One feature of the PC versions of RPG Maker programs is that a user can create new tilesets and characters, and add any new graphics the user wants. Despite being geared towards creating role-playing video games, the engine also has the capability to create games of other genres, such as adventure games, story-driven games or visual novels with minimal tweaking.

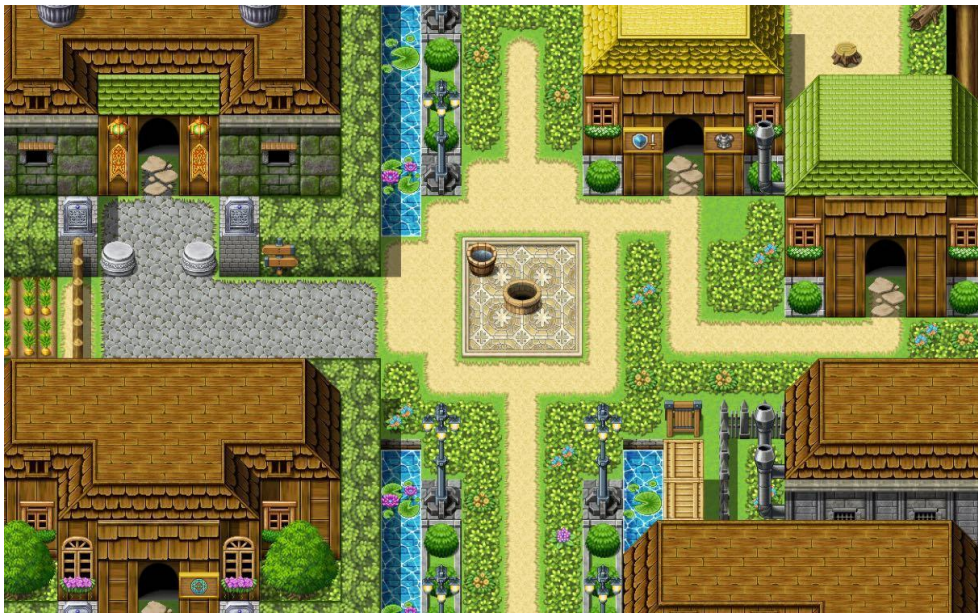


Figure 34 A game created with RPG Maker

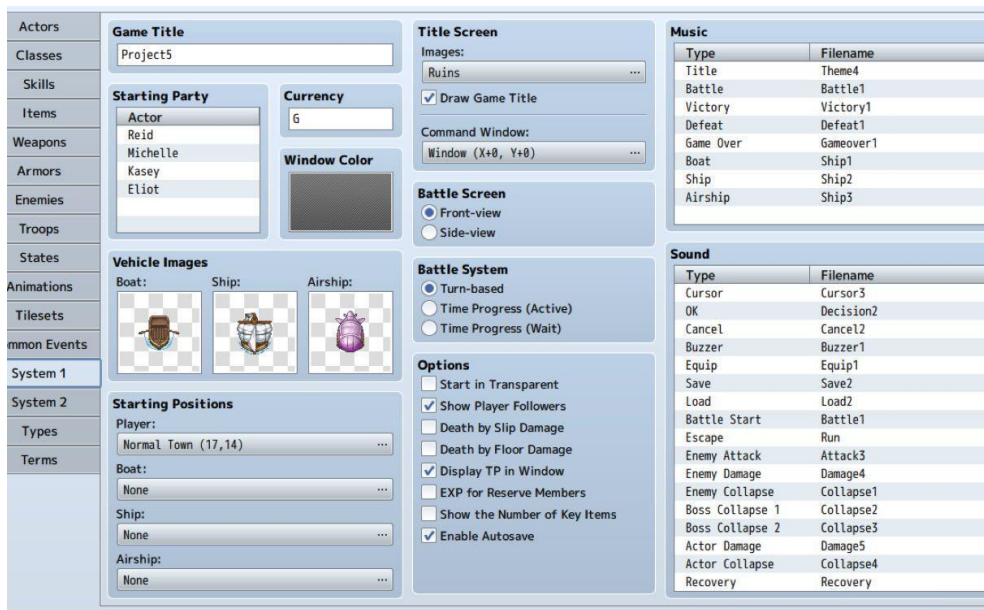


Figure 35 The RPG Maker interface

CONSTRUCT

Construct's block-based approach is a simple way to start designing games. There's no need to learn the syntax of complicated programming languages. Each block is a list of conditions on the left. When those conditions are met, it performs the actions on the right.

Construct 3 has an event-based system that allows to learn the principles of programming in an accessible and easy to understand way. Construct 3 is the fastest way to get a game up and running using drag and drop tools and features. Advanced students can extend Construct 3 by writing their own plugins and behaviors in Javascript. Construct 3 runs in the browser and automatically updates to the latest version.

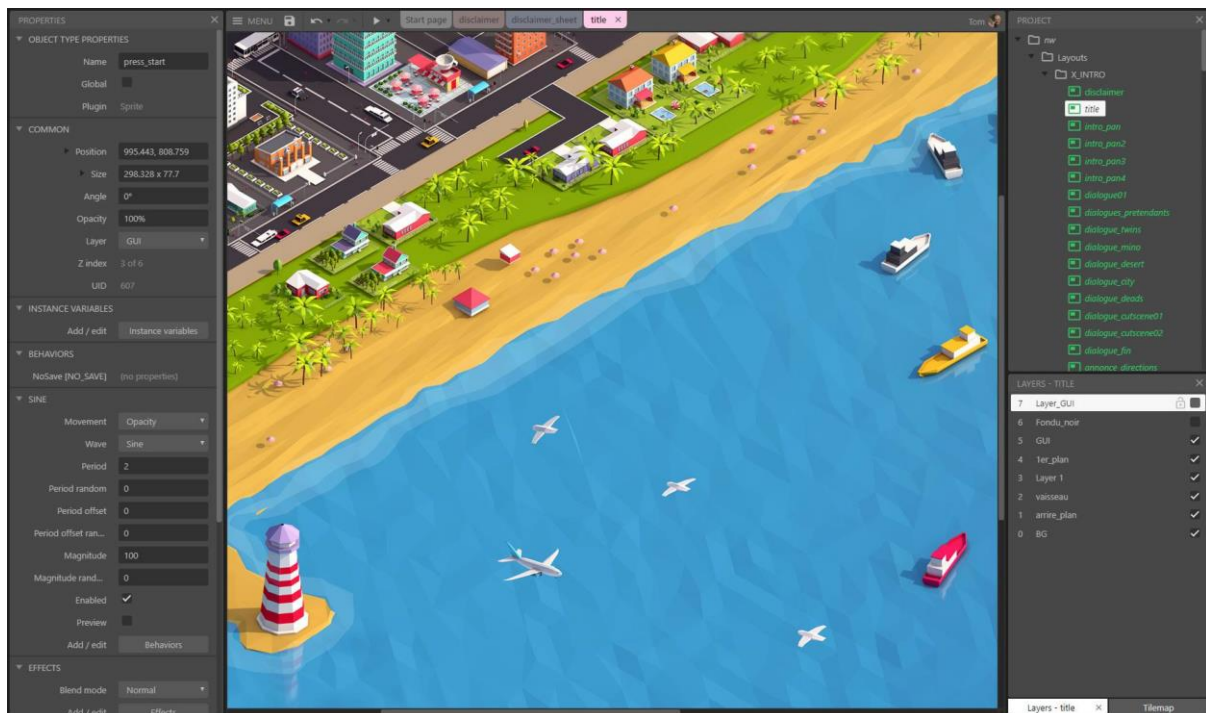


Figure 36 The Construct interface

GDEVELOP

GDevelop is a free, open-source, and cross-platform game creation tool that anyone can use to create games without programming skills. GDevelop is mainly aimed at non-programmers and game developers of all skillsets, employing event based visual programming similar to

such engines as Construct. GDevelop is available for all major operating systems, including the latest versions of Windows, macOS, and Linux.

In GDevelop games are created without using any programming languages, through three methods:

- **Event-based Logic:** the Event system creates logic by monitoring for Conditions on when to trigger, and actions to take once the event conditions are met. The majority of events are presented in normalized language, so creators can avoid having to understand coding concepts found in many programming languages.
- **Behaviors:** Behaviors allow for advanced combinations of pre-built functions and events to add logic like physics-based movement, pathfinding, acting as a platform or platform character game, allowing to move the object with the mouse or touch, transitions, etc. Behaviors can be added to game objects, and the same object can have several behaviors. Behaviors can also be created using the Event system - allowing users to extend the existing set of behaviors without coding.
- **Easy Content Pipeline:** All game content, such as character art, backgrounds, text, etc, can be added directly through a point and click interface in the editor. Some example content types are Sprites, Tiled Sprites, 9-Patch (Panel) Sprites, Text Objects, Text Objects with BBText support, Shape Painters, and more. Music and Sounds can be imported directly into the events that utilize them.



Figure 37 A game developed with GDevelop

STENCYL

Stencyl is a video game development tool that allows users to create 2D video games for computers, mobile devices, and the web. The software is available for free, with select publishing options available for purchase. Stencyl includes an intuitive toolset that accelerates the game development workflow. Stencyl supports iOS (iPhone/iPad), Android, Windows, Mac, Linux, HTML5.

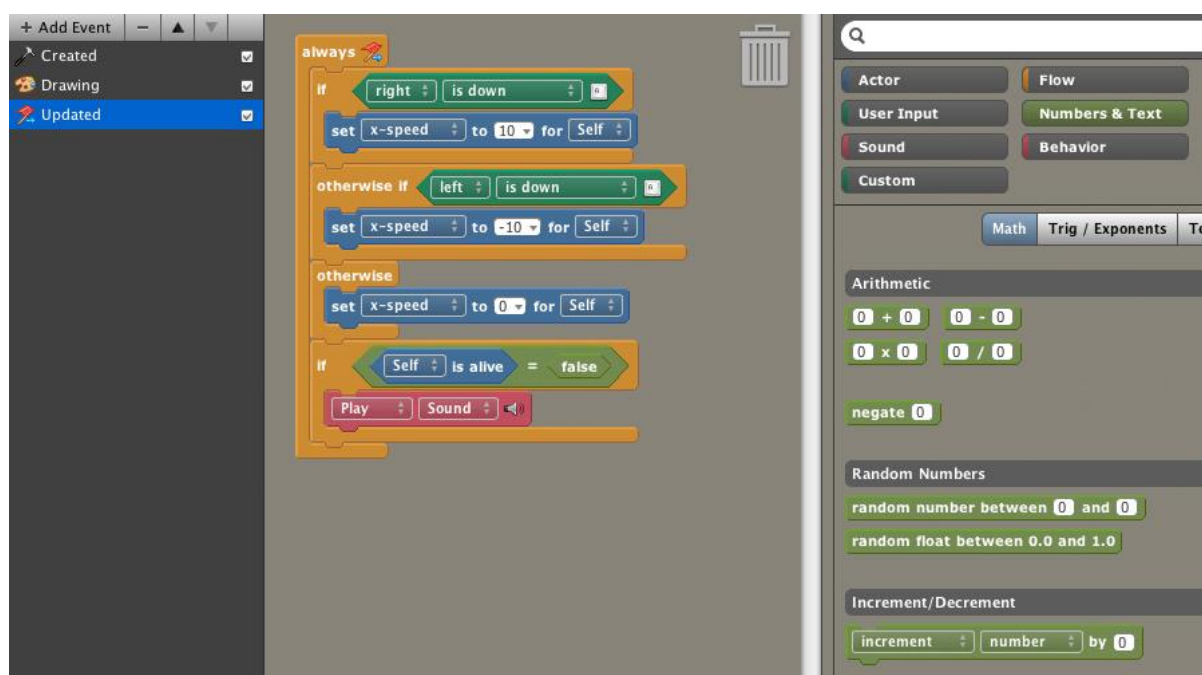


Figure 38 The Stencyl interface

O1/A5 - PROGRAMMING SKILL BUILDING REQUIREMENTS FOR ADULT EDUCATION

Based on the outputs from the previous tasks, this task focused on the definition of learning requirements for building programming skills among adults. The work documents learning objectives, learning outcomes, and methods for assessing the educational added value introduced by the proposed serious games methodology for digital skill building.

The reference frameworks which have been used in the redaction of this document are the following:

- [DigComp | EU Science Hub](#)
- [European Competence Framework](#)
- [Home Essential digital skills framework](#)

The scenarios proposed by the SILVERCODERS project will be related to the creative sector (game design, web design, multimedia, social media information provider, etc.).

The SILVERCODERS project's serious games methodology for digital skill building is aims to produce two types of expected learning outcomes:

- General digital competencies
- Programming-related digital competencies

Programming-related competencies are learning objectives that cover a very specific coding concept. Those concepts range from basic to advanced ones and will be covered by the SILVERCODERS challenges in a progressing fashion, each piece of knowledge opening the way to the understanding and acquiring of the next one.

General digital competencies are not specific to programming and cover all the aspects a citizen today has to know to evolve in the digital world. Those competencies will be covered by a few of the SILVERCODERS challenges, but will mainly be worked on and acquired in a transversal manner, as they will be called upon and needed in the practice of acquiring programming related digital competencies.

Expected general digital competencies

In accordance with the DigiComp 2.0 model, the following competencies are expected to be covered by the SILVERCODERS project.

Type of Competency	Detailed list
1. Information and data literacy	<p>1.1. Browsing, searching and filtering data, information and digital content To articulate information needs, to search for data, information and content in digital environments, to access them and to navigate between them. To create and update personal search strategies.</p> <p>1.2. Evaluating data, information and digital content To analyse, compare and critically evaluate the credibility and reliability of sources of data, information and digital content. To analyse, interpret and critically evaluate the data, information and digital content.</p> <p>1.3. Managing data, information and digital content To organise, store and retrieve data, information and</p>

	content in digital environments. To organise and process them in a structured environment.
2. Communication and collaboration	<p>2.1. Interacting through digital technologies To interact through a variety of digital technologies and to understand appropriate digital communication means for a given context.</p> <p>2.2. Sharing through digital technologies To share data, information and digital content with others through appropriate digital technologies. To act as an intermediary, to know about referencing and attribution practices.</p> <p>2.3. Engaging in citizenship through digital technologies To participate in society through the use of public and private digital services. To seek opportunities for self-empowerment and for participatory citizenship through appropriate digital technologies.</p> <p>2.4. Collaborating through digital technologies To use digital tools and technologies for collaborative processes, and for co-construction and co-creation of resources and knowledge.</p> <p>2.5. Netiquette To be aware of behavioural norms and know-how while using digital technologies and interacting in digital environments. To adapt communication strategies to a specific audience and to be aware of cultural and generational diversity in digital environments.</p> <p>2.6. Managing digital identity To create and manage one or multiple digital identities, to be able to protect one's own reputation, to deal with the data that one produces through several digital tools, environments and services.</p>
3. Digital content creation	<p>3.1. Developing digital content To create and edit digital content in different formats, to express oneself through digital means.</p> <p>3.2. Integrating and re-elaborating digital content To modify, refine, improve and integrate information and content into an existing body of knowledge to create new, original and relevant content and knowledge.</p> <p>3.3. Copyright and licences To understand how copyright and licences apply to data, information and digital content.</p> <p>3.4. Programming To plan and develop a sequence of understandable</p>

	instructions for a computing system to solve a given problem or perform a specific task.
4. Safety	<p>4.1. Protecting devices To protect devices and digital content, and to understand risks and threats in digital environments. To know about safety and security measures and to have due regard to reliability and privacy.</p> <p>4.2. Protecting personal data and privacy To protect personal data and privacy in digital environments. To understand how to use and share personally identifiable information while being able to protect oneself and others from damages. To understand that digital services use a “Privacy policy” to inform how personal data is used.</p> <p>4.3. Protecting health and well-being To be able to avoid health-risks and threats to physical and psychological well-being while using digital technologies. To be able to protect oneself and others from possible dangers in digital environments (e.g. cyber bullying). To be aware of digital technologies for social well-being and social inclusion.</p> <p>4.4. Protecting the environment To be aware of the environmental impact of digital technologies and their use.</p>
5. Problem solving	<p>5.1. Solving technical problems To identify technical problems when operating devices and using digital environments, and to solve them (from trouble-shooting to solving more complex problems).</p> <p>5.2. Identifying needs and technological responses To assess needs and to identify, evaluate, select and use digital tools and possible technological responses to solve them. To adjust and customise digital environments to personal needs (e.g. accessibility).</p> <p>5.3. Creatively using digital technologies To use digital tools and technologies to create knowledge and to innovate processes and products. To engage individually and collectively in cognitive processing to understand and resolve conceptual problems and problem situations in digital environments.</p> <p>5.4. Identifying digital competence gaps To understand where one’s own digital competence needs to be improved or updated. To be able to support others with their digital competence development. To seek opportunities for self-development and to keep up-to-date with the digital evolution.</p>

Expected programming related competencies

The following list of programming competencies has been established.

Type of Competency	Detailed list
1. Programming Languages and Platforms	<p>1.1. Compilers and Tools To understand how code is treated by a computer and what is the role of a compiler.</p> <p>1.2. Low and High level languages To be familiar with the concept of low and high level languages and understand what their differences are and what is required to code in either of them.</p> <p>1.3. Visual Programming To have experience with a visual programming suite and be able to code standard small piece of software with it.</p> <p>1.4. No code Programming To have knowledge of the concept of no code programming and understand all the advantages and limitations of such solutions.</p>
2. Expressions and Statements	<p>2.1. Command Lines Know what statements and command lines are and what they mean for a compiler.</p> <p>2.2. Syntax To be able to write instructions using correct syntax and with minimal errors.</p> <p>2.3. Operators Know what operators are, what they do and which symbols stand for which operators.</p> <p>2.4. Input and Output Understand the concepts of inputs and how they can modify what a program is going to output.</p> <p>2.5. Comments To understand the importance of commenting code, to have the knowledge of writing comments and the discipline to do it often.</p>
3. Variables	<p>3.1. Variables To be able to understand the assignment of values to variables and how to change them.</p> <p>3.2. Constants To know how and when to use constants instead of variables.</p>

	<p>3.3. Reserved Words To be able to identify and recognize reserved words in different programming languages and know how to use them.</p>
4. Mathematical operations	<p>4.1. Basic Operations To know all the basic arithmetic operations and how to use them.</p> <p>4.2. Trigonometry To be able to apply more complex trigonometric calculations such as sinus, cosine and tangents.</p> <p>4.3. Advanced Operations Recognize and know how to use more complex operations such as modulo, factorial and powers.</p> <p>4.4. Random To be able to integrate random numbers into code and to understand what the limitations of pseudo-randomness are.</p>
5. Data Structures	<p>5.1. Numbers Recognize and know how to use all the data structures related to numbers. Being able to know the differences between them and why some are more adapted than others in certain situations.</p> <p>5.2. Strings To know the structures linked to the use of text, such as strings and characters. To be able to use special characters and be aware of the issues with non latin characters.</p> <p>5.3. Lists To be able to use lists in order to store collections of objects and to know special operations that can be used on them such as sorting.</p> <p>5.4. Arrays To be able to use arrays in order to store collections of numbers and to know special operations that can be used on them.</p> <p>5.5. Media To be able to operate with media (audio, video, images, etc.) structures.</p>
6. Control Structures	<p>6.1. Functions To know and understand how to use functions to organize the code and avoid code repetition and improve reusability</p> <p>6.2. Conditionals To be able to use If and Switch statements correctly to execute code according to a certain defined fixed condition.</p>

	<p>To be able to write imbricated conditionals in order to treat complex issues.</p> <p>6.3. Loops</p> <p>To know how to use loops to treat a certain situation many times. Being able to write correct conditions for starting and stopping loops and avoid infinite loops.</p>
7. Object Oriented Programming Concepts	<p>7.1. Object Oriented Languages</p> <p>To know and understand the programming paradigm based on the concepts of objects containing data and code.</p> <p>7.2. Objects and Classes</p> <p>To be familiar with the concept of classes as a definition for a data format and all the available procedures to modify them. Being able to understand how objects are instances of classes.</p> <p>7.3. Class and prototypes</p> <p>To know and understand the differences between class-based languages and prototype-based languages and their respective advantages and drawbacks.</p> <p>7.4. Encapsulation</p> <p>How to use encapsulation to bind together data and functions that manipulate the data and keep them both safe from outside interference.</p> <p>7.5. Composition, Inheritance and Delegation</p> <p>To be familiar with the concepts of objects containing other objects in the instance variables (composition), with how classes can be arranged in a hierarchy that represents relationships (inheritance) and how one entity can pass something to another (delegation).</p> <p>7.6. Polymorphism</p> <p>Being able to recognize and create code that can be agnostic as to which class it is operating on.</p> <p>7.7. Open Recursion</p> <p>To know that object methods can call other methods of the same object and to be able to recognize keywords such as <i>this</i>.</p>
8. Debugging	<p>8.1. Debugging Tools</p> <p>To know sets of tools which can be useful in order to help remove the bugs of a certain piece of code.</p> <p>8.2. Debugging Methodologies</p> <p>To be able to debug code written by someone else and to be familiar with common errors and mistakes in code writing.</p>

O1/A6 - SKILL DEVELOPMENT REQUIREMENTS FOR ADULT TRAINERS

This task focused on an analysis of skill development requirements for adult trainers in relation to introducing innovative practices in their classroom for promoting programming skill development. This aims at promoting the long-term motivation of trainers in their professional development through their own skill enhancement and enrichment of their instructional practices as a career satisfaction strategy as a result of lifelong training. Already existing skills of trainers were taken into account and a research regarding the possible upgrade of these skills was conducted in order to see how one can facilitate the integration of innovative technology into their instructional practices. A particular focus was dedicated to the creative sector skills and abilities considering the functional requirements for occupations in game design, multimedia, content production, social media production and analysis, video and audio production.

Each partner involved in this particular activity interviewed 3-5 trainers with help of a survey that was developed to get a deeper understanding of the professional and pedagogical skills required by a programming teacher.

In Italy, the specific investigation on the training skills involved 4 trainers on ICTs and coding coming from several backgrounds: private sector, training centres, non-profit organizations.

The respondents highlighted various skills, both pedagogical and professional, that a trainer should have while promoting programming skill development for adults 55+. From the prioritization of those skills emerge some common strands among the answers: clear communication abilities; patience; keeping the language simple and starting from very basic concepts; ability to prepare feedback sessions on the modules with the participants; organization and method; clarity in the structure of the training programme; ability to prepare practice session and working proofs of concept with the participants.

Regarding innovation in promoting programme skills development among adults 55+, the respondents suggested several approaches such as: using learning apps for smartphones and

tablets; adopting the learning by doing formula while organizing targeted workshops; using an informal and playful approach.

For what concerns the possible integration of the existing skills of a trainer with new programming skills related to the creative sector, among the respondents was stressed the need of combining technological and artistic contents through specific methods like the Steam Education. Complete knowledge of the kind of projects that are normally implemented in a specific creative sector, together with a deep understanding of how they are usually developed, was also suggested as a way to guarantee skills integration.

In Romania, the 2 questionnaires were applied to 2 university teachers from the West University of Timisoara, these being teachers of ICT and e-Pedagogy and 3 adult trainers who develop educational services for different target groups from young people to seniors. The results following the application of these questionnaires showed us the educational need to update the competency profile of the trainer, the adult trainer, within this constantly moving and changing society.

In addition to traditional skills that the trainer must have and constantly improve, such as: good communication, empathy, social skills, adaptability, problem solving, promoting student accessibility, can be seen their emerging need for open education, implicitly an open pedagogy and assisted by the new technology of communication and teaching through the development of creative capacities and abilities to create visual, audio and text content by coding in order to create teaching materials and tools to support the teaching-learning process. Therefore, there is an acute need highlighted by the development of these creative and multimedia skills, in addition to traditional ICT skills. A better reporting of these collected educational needs of the trainers can be seen below in figure 1 representing the mapping of all traditional competencies and new updating trends.

In Sweden, the answers to the questionnaire regarding specific training skills were collected from 3 trainers who previously participated in the initial survey of this intellectual output. The participants were asked to list 3 most crucial pedagogical and professional skills. All trainers

listed similar skills for the pedagogical part such as: clear communications towards the learners, motivation and having fun as well as being able to explain complex context in a simple way and with reference to things learners are already familiar with. The most important professional skills were listed as having broad experience both from various programming programs but also in life. All of them agreed that being able to motivate the learners in creative ways was the most crucial skill in order to get people above the age of 55 engaged in coding. The respondents believed it to be important to implement aspects of instant feedback so the learner gets a positive experience which will lead to further motivation. The implementation of games is also a good way to engage first time learners in a fun and pedagogical way in order to capture their interest according to their answers. Overall, they believe it to be very important to be able to keep the learners' motivation high and through encouragement and constant feedback in ways of creative solutions they think that to be possible.

The main objective of a trainer is to create training programs that develop certain skills and improve the knowledge of the participants in the field that is the theme of the course. The training programs can be closed, respectively addressed only to a certain organization, or open, addressed to all persons interested in the respective topic. From a professional point of view, the trainer must have the ability to plan, organize and execute the training process, in order to be able to implement the themes and objectives of the session, in an experiential manner. That is why the profile of the trainer is defined by his pedagogical and professional competences and abilities (in a certain field) in this case and of the research undertaken within the SILVERCODERS project, refers to those necessary competencies in the creative sector, which can easily be transposed and correlated between the two categories.

The data analysed following the study carried out within the SILVERCODERS project reminded of those traditional competencies that a trainer should have developed, as well as, we mention:

- strong communication by adapting the language according to the group of beneficiaries, to know how to communicate with others

- problem solving and creativity- Ability to think & identify a solution to a complex situation & problem, using imagination or original idea.
- supporting learners - Support & motivate groups of learner, learning communities; Promote the development of learners autonomy / confidence & effectiveness in a supportive learning environment

There is nothing wrong with these roles and responsibilities. But new training methods, new technologies, new ways of building competencies and new expectations of trainees create the gap between what is perceived as a good set of trainer's competencies and what is demanded by the market.

Every modern trainer should be ready to have skills in the creative sector which means he is able to develop infographics, audio-visual materials, multimedia content and knows programming to create an application, a game, a web page, etc.

Also, the role of content curator is important. There are many pieces of valuable content available - some of them could be used by trainees as a pre-work, stimulus for a reflection, implementation tool, etc. To curate content a trainer should be ready to search, choose, describe, validate and update his/her selection. Such activities require a good understanding of Internet space in terms of search engines usage, IP issues, communities of practice which are available, spaces of open content, etc.

Also, the new ability to edit different materials: text, audio, visual is closely correlated with its ability to adapt the content and promote equal opportunities for those students who have different educational needs, which is found in pedagogical skills and traditional.

The constant professional development is closely related to the professional competence to fits the method to learning situation, especially when we talk about a blended learning.

In conclusion, it can be said that in terms of pedagogical skills, clear communication, patience and motivational treats are important to engage adults of the age 55+ in coding. Professional skills that topped the list were extensive ICT skills, being commutative and life experience.

Regarding innovation in promoting programme skills development among adults 55+ it seems like games, workshops and feedback sessions seem to play an important role. Proof of concept was also mentioned by the trainers as an important feedback loop to the learners.

From all the data analysed following the research undertaken by the SILVERCODERS consortium emerges the acute need to redefine the competency profile of the trainer, adult educator and equip him/her with new competitive competencies in the socio-economic life of the world. According to a large European Commission study on Adult Learners in Digital Learning Environments (2015), it is shown that:

- Educators need digital and pedagogical skills. Trainers must be provided with training in the effective use of ICT and in the creative and coding sector and be fully involved in the design of programs;
- The benefits of adult learning are not effectively communicated and understood. Communicating the particular benefits of ICT-based learning can better motivate adults to learn and help adults and companies understand the rewards and benefits of adult learning;
- Learning providers need a wider network, exchange of best practices and partnerships to create high-quality, targeted learning content for ICT, creative sector, digital and coding, capable of creating multimedia, visual, audio-video content, web graphics etc. for their adult students.

So, on the one hand, we are witnessing the issue of new technologies, and on the other hand, with this global crisis caused by COVID-19, which has also impacted the field of adult education, at this time, have made the development of infrastructure and adaptability human capital, to accelerate rapidly and converge towards the development and implementation of new techniques and teaching strategies adapted to the context, a particular focus must be dedicated to the creative sector skills and abilities considering the functional requirements for occupations in game design, multimedia, content production, social media production and analysis, video and audio production.

SILVERCODERS LEARNING FRAMEWORK

The overall objective of Output 1, concretized through this task, is to develop a learning framework that enhances learning experiences for adult education in relation to building programming skills. The framework exposes learners to approaches that are well accepted for designing and implementing products and services that effectively meet end user needs. The proposed framework adapts design thinking approaches, which are inherently user-centered, to ICT learning practices. The approach is in-line with software engineering practices that start by challenging professionals and learners to address needs before designing the details of an effective solution. Gamification approaches will complement the proposed framework by encouraging learners to roleplay by simulating practices that learners will be exposed to. The specification and design of the methodological pedagogical framework describes the training model including requirements analysis, methodology definition, learning strategies, support technologies, learning moments, etc. Once the project has finished the pedagogical model will be revised to incorporate the corrective measures identified along the implementation and can then be made available to other European adults training centres interested in exploring the model proposed.

REQUIREMENTS' ANALYSIS

The analysis of the stakeholders reached the following general description of the requirements:

1. To provide adults with the necessary tools and competencies to develop creative and innovative solutions to face new risks and challenges, both in personal, educational and professional contexts.
2. To target the creative and cultural sectors to support them becoming more digital and modern, more adaptable, resilient, and able to survive and prosper in the current situation and possible future challenges as well.

The complete list of competences selected for the SILVERCODERS learning framework was the following:

Type of Competency	Detailed list
1. Information and data literacy	<p>1.1. Browsing, searching and filtering data, information and digital content To articulate information needs, to search for data, information and content in digital environments, to access them and to navigate between them. To create and update personal search strategies.</p> <p>1.2. Evaluating data, information and digital content To analyse, compare and critically evaluate the credibility and reliability of sources of data, information and digital content. To analyse, interpret and critically evaluate the data, information and digital content.</p> <p>1.3. Managing data, information and digital content To organise, store and retrieve data, information and content in digital environments. To organise and process them in a structured environment.</p>
2. Communication and collaboration	<p>2.1. Interacting through digital technologies To interact through a variety of digital technologies and to understand appropriate digital communication means for a given context.</p> <p>2.2. Sharing through digital technologies To share data, information and digital content with others through appropriate digital technologies. To act as an intermediary, to know about referencing and attribution practices.</p> <p>2.3. Engaging in citizenship through digital technologies To participate in society through the use of public and private digital services. To seek opportunities for self-empowerment and for participatory citizenship through appropriate digital technologies.</p> <p>2.4. Collaborating through digital technologies To use digital tools and technologies for collaborative processes, and for co-construction and co-creation of resources and knowledge.</p> <p>2.5. Netiquette To be aware of behavioural norms and know-how while using digital technologies and interacting in digital environments. To adapt communication strategies to a</p>

	<p>specific audience and to be aware of cultural and generational diversity in digital environments.</p> <p>2.6. Managing digital identity To create and manage one or multiple digital identities, to be able to protect one's own reputation, to deal with the data that one produces through several digital tools, environments and services.</p>
3. Digital content creation	<p>3.1. Developing digital content To create and edit digital content in different formats, to express oneself through digital means.</p> <p>3.2. Integrating and re-elaborating digital content To modify, refine, improve and integrate information and content into an existing body of knowledge to create new, original and relevant content and knowledge.</p> <p>3.3. Copyright and licences To understand how copyright and licences apply to data, information and digital content.</p> <p>3.4. Programming To plan and develop a sequence of understandable instructions for a computing system to solve a given problem or perform a specific task.</p>
4. Safety	<p>4.1. Protecting devices To protect devices and digital content, and to understand risks and threats in digital environments. To know about safety and security measures and to have due regard to reliability and privacy.</p> <p>4.2. Protecting personal data and privacy To protect personal data and privacy in digital environments. To understand how to use and share personally identifiable information while being able to protect oneself and others from damages. To understand that digital services use a "Privacy policy" to inform how personal data is used.</p> <p>4.3. Protecting health and well-being To be able to avoid health-risks and threats to physical and psychological well-being while using digital technologies. To be able to protect oneself and others from possible dangers in digital environments (e.g. cyber bullying). To be aware of digital technologies for social well-being and social inclusion.</p> <p>4.4. Protecting the environment To be aware of the environmental impact of digital technologies and their use.</p>
5. Problem solving	<p>5.1. Solving technical problems To identify technical problems when operating devices and</p>

	<p>using digital environments, and to solve them (from trouble-shooting to solving more complex problems).</p> <p>5.2. Identifying needs and technological responses To assess needs and to identify, evaluate, select and use digital tools and possible technological responses to solve them. To adjust and customise digital environments to personal needs (e.g. accessibility).</p> <p>5.3. Creatively using digital technologies To use digital tools and technologies to create knowledge and to innovate processes and products. To engage individually and collectively in cognitive processing to understand and resolve conceptual problems and problem situations in digital environments.</p> <p>5.4. Identifying digital competence gaps To understand where one's own digital competence needs to be improved or updated. To be able to support others with their digital competence development. To seek opportunities for self-development and to keep up-to-date with the digital evolution.</p>
6. Programming Languages and Platforms	<p>6.1 Compilers and Tools To understand how code is treated by a computer and what is the role of a compiler.</p> <p>6.2 Low and High level languages To be familiar with the concept of low and high level languages and understand what their differences are and what is required to code in either of them.</p> <p>6.3 Visual Programming To have experience with a visual programming suite and be able to code standard small piece of software with it.</p> <p>6.4 No code Programming To have knowledge of the concept of no code programming and understand all the advantages and limitations of such solutions.</p>
7. Expressions and Statements	<p>7.1. Command Lines Know what statements and command lines are and what they mean for a compiler.</p> <p>7.2. Syntax To be able to write instructions using correct syntax and with minimal errors.</p> <p>7.3. Operators Know what operators are, what they do and which symbols stand for which operators.</p>

	<p>7.4. Input and Output Understand the concepts of inputs and how they can modify what a program is going to output.</p> <p>7.5. Comments To understand the importance of commenting code, to have the knowledge of writing comments and the discipline to do it often.</p>
8. Variables	<p>8.1. Variables To be able to understand the assignment of values to variables and how to change them.</p> <p>8.2. Constants To know how and when to use constants instead of variables.</p> <p>8.3. Reserved Words To be able to identify and recognize reserved words in different programming languages and know how to use them.</p>
9. Mathematical operations	<p>9.1. Basic Operations To know all the basic arithmetic operations and how to use them.</p> <p>9.2. Trigonometry To be able to apply more complex trigonometric calculations such as sinus, cosine and tangents.</p> <p>9.3. Advanced Operations Recognize and know how to use more complex operations such as modulo, factorial and powers.</p> <p>9.4. Random To be able to integrate random numbers into code and to understand what the limitations of pseudo-randomness are.</p>
10. Data Structures	<p>10.1. Numbers Recognize and know how to use all the data structures related to numbers. Being able to know the differences between them and why some are more adapted than others in certain situations.</p> <p>10.2. Strings To know the structures linked to the use of text, such as strings and characters. To be able to use special characters and be aware of the issues with non latin characters.</p> <p>10.3. Lists To be able to use lists in order to store collections of objects and to know special operations that can be used on them such as sorting.</p>

	<p>10.4. Arrays To be able to use arrays in order to store collections of numbers and to know special operations that can be used on them.</p> <p>10.5. Media To be able to operate with media (audio, video, images, etc.) structures.</p>
11. Control Structures	<p>11.1. Functions To know and understand how to use functions to organize the code and avoid code repetition and improve reusability</p> <p>11.2. Conditionals To be able to use If and Switch statements correctly to execute code according to a certain defined fixed condition. To be able to write imbricated conditionals in order to treat complex issues.</p> <p>11.3. Loops To know how to use loops to treat a certain situation many times. Being able to write correct conditions for starting and stopping loops and avoid infinite loops.</p>
12. Object Oriented Programming Concepts	<p>12.1. Object Oriented Languages To know and understand the programming paradigm based on the concepts of objects containing data and code.</p> <p>12.2. Objects and Classes To be familiar with the concept of classes as a definition for a data format and all the available procedures to modify them. Being able to understand how objects are instances of classes.</p> <p>12.3. Class and prototypes To know and understand the differences between class-based languages and prototype-based languages and their respective advantages and drawbacks.</p> <p>12.4. Encapsulation How to use encapsulation to bind together data and functions that manipulate the data and keep them both safe from outside interference.</p> <p>12.5. Composition, Inheritance and Delegation To be familiar with the concepts of objects containing other objects in the instance variables (composition), with how classes can be arranged in a hierarchy that represents relationships (inheritance) and how one entity can pass something to another (delegation).</p>

	<p>12.6. Polymorphism Being able to recognize and create code that can be agnostic as to which class it is operating on.</p> <p>12.7. Open Recursion To know that object methods can call other methods of the same object and to be able to recognize keywords such as <i>this</i>.</p>
13. Debugging	<p>13.1. Debugging Tools To know sets of tools which can be useful in order to help remove the bugs of a certain piece of code.</p> <p>13.2. Debugging Methodologies To be able to debug code written by someone else and to be familiar with common errors and mistakes in code writing.</p>

LEARNING STRATEGY

The selected competences were organized in eight modules with the following organization:

Module	Competences
1. Basic digital literacy	Information and data literacy Communication and collaboration Safety
2. No-code app creation	Information and data literacy Digital content creation
3. Digital content creation	Digital content creation
4. Programming and coding	Programming Languages and Platforms Expressions and Statements Variables Mathematical operations
5. Data Structures	Data Structures
6. Control structures	Control Structures
7. Advanced coding	Object Oriented Programming Concepts Debugging
8. Problem solving	Problem solving

The SILVERCODERS Learning Strategy is based on the following elements:

Element	Purpose
Classroom sessions	Presentation of concepts. Supported practice. Scheduled group activities.
Autonomous learning	Autonomous practice. Unscheduled individual activities.
Video-lectures, demonstrations or simulations	Presentation of concepts. Can support autonomous learning or classroom sessions.
Half-baked scenarios or challenges	Practice activities. Can support autonomous learning or classroom sessions.
Assessment	Self-assessment activities.

The SILVERCODERS Learning Strategy adopts a blended-learning approach with the following organization:

Element	Purpose
Workload	4-8 hours per week
Module	1 module per week
Classroom sessions	1 or 2 2-hour session per week
Autonomous learning	2-6 hours per week
Video-lectures, demonstrations or simulations	2-4 video-lectures/demos per module
Half-baked scenarios or challenges	4 challenges per module
Assessment	4 self-assessment activities per module 1 assessment activity per module

REFERENCES

Alice - Tell Stories. Build Games. Learn to Program. Retrieved from <http://www.alice.org/>

Alserri, S.A., et al. (2018). Gender-based Engagement Model for Serious Games. International Journal on Advanced Science, Engineering and Information Technology, 8(4), 1350-1357. doi:10.18517/ijaseit.8.4.6490

AppInventor, With MIT App Inventor, anyone can build apps with global impact. Retrieved from <http://appinventor.mit.edu/>

Bartilla, A.C. Köppe. (2015). Awareness seeds for more gender diversity in computer science education. Paper presented at the Proceedings of the 20th European Conference on Pattern Languages of Programs, Kaufbeuren, Germany

Carmichael, G. (2008). Girls, computer science, and games. ACM SIGCSE Bulletin, 40(4), 107-110. doi:10.1145/1473195.1473233

CodeCombat - Learn how to code by playing a game. Retrieved from <https://codecombat.com/>

Coding for Kids | Tynker. Retrieved from <https://www.tynker.com/>

Combéfis, S., et al. (2016). Learning Programming through Games and Contests: Overview, Characterisation and Discussion. Olympiads in Informatics, 10, 39-60. doi:10.15388/ioi.2016.03

Cooper, S., et al. (2000). Alice: a 3-D tool for introductory programming concepts. Journal of Computing Sciences in Colleges, 15(5), 107-116.

Dunand M., Developing Visual Accessibility Options to Empower Grade School Students in Designing Inclusive Mobile Applications, M.Eng thesis, Elect. Eng. Comput. Sci., Massachusetts Inst. of Technol., Cambridge, 2021



European Commission. (2014, Julho 18). Coding—The 21st century skill. Shaping Europe's Digital Future - European Commission. <https://ec.europa.eu/digital-single-market/en/coding-21st-century-skill>

Franković, I., et al. (2018). Serious Games for Learning Programming Concepts. Paper presented at the 8th International Conference the Future of Education, Florence, Italy

Freitas, S.S. Jarvis. (2007). Serious games—engaging training solutions: A research and development project for supporting training needs. 38(3), 523-525. doi:10.1111/j.1467-8535.2007.00716.x

Geist, E. (2016). Robots, Programming and Coding, Oh My! Childhood Education, 92(4), 298-304. doi:10.1080/00094056.2016.1208008

Grivokostopoulou, F., et al. (2016). An Educational Game for Teaching Search Algorithms. Proceedings of the 8th International Conference on Computer Education 2, 129-136. doi:10.5220/0005864601290136

Hosein, A. (2019). Girls' video gaming behaviour and undergraduate degree selection: A secondary data analysis approach. Computers in Human Behaviour, 91, 226-235. doi:10.1016/j.chb.2018.10.001

Hsu T., Hal Abelson, Jessica Van Brummelen, (2021). The Effects of Applying Experiential Learning into the Conversational AI Learning Platform on Secondary School Students

Hsu T., Hal Abelson, Natalie Lao, Yu-Han Tseng, (2021). Behavioral-Pattern Exploration and Development of An Instructional Tool for Young Children to Learn AI

Jemmali, C. (2016). May's Journey: A serious game to teach middle and high school girls programming. Worcester Polytechnic Institute, Retrieved from <https://digitalcommons.wpi.edu/etd-theses/455>

Johnson, C., et al. (2016). Game Development for Computer Science Education. Paper presented at the Proceedings of the 2016 ITiCSE Working Group Reports, Arequipa, Peru

Kafai, Y.B. (2006). Playing and Making Games for Learning: Instructionist and Constructionist Perspectives for Game Studies. Games and Culture, 1(1), 36-40. doi:10.1177/1555412005281767

Kazimoglu, C., et al. (2012). A Serious Game for Developing Computational Thinking and Learning Introductory Computer Programming. Procedia - Social and Behavioral Sciences, 47, 1991-1999. doi:10.1016/j.sbspro.2012.06.938

Kelleher, C., et al. (2007). Storytelling alice motivates middle school girls to learn computer programming. Paper presented at the Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, San Jose, California, USA

LightBot Retrieved from <http://lightbot.com/>

Malliarakis, C., et al. (2014). CMX: Implementing an MMORPG for learning programming. Paper presented at the 8th European Conference on Games Based Learning, Berlin

Mathrani, A., et al. (2016). PlayIT: Game Based Learning Approach for Teaching Programming Concepts. Educational Technology & Society, 19(2), 5-17.

Meerbaum-Salant, O., et al. (2013). Learning computer science concepts with Scratch. Computer Science Education, 23(3), 239-264. doi:10.1080/08993408.2013.832022

Michael, D.S. Chen. (2006). Serious Games: Games That Educate, Train and Inform. Canada: Thomson Course Technology PTR.

Miljanovic, M.A.J.S. Bradbury. (2016). Robot on!: a serious game for improving programming comprehension. Paper presented at the Proceedings of the 5th International Workshop on Games and Software Engineering, Austin, Texas

Miljanovic, M.A.J.S. Bradbury. (2018, November 7-8, 2018). A Review of Serious Games for Programming. Paper presented at the 4th Joint International Conference on Serious Games, Darmstadt, Germany.

Oblinger, D.G. (2004). The next generation of educational engagement. *Journal of Interactive Media in Education*(8), 1-18. doi:10.5334/2004-8-oblinger

Oblinger, D.G. (2006). Simulations, Games, and Learning [Online]. Retrieved from <http://net.educause.edu/ir/library/pdf/ELI3004.pdf>

Osborn, G. (2017). Male and Female Gamers: How Their Similarities and Differences Shape the Games Market. Retrieved from <https://newzoo.com/insights/articles/male-and-female-gamers-how-their-similarities-and-differences-shape-the-games-market/>

Ouahbi, I., et al. (2015). Learning Basic Programming Concepts by Creating Games with Scratch Programming Environment. *Procedia - Social and Behavioral Sciences*, 191, 1479-1482. doi:10.1016/j.sbspro.2015.04.224

Prensky, M. (2003). Digital game-based learning %J *Comput. Entertain.* 1(1), 21-21. doi:10.1145/950566.950596

Resnick, M., et al. (2009). Scratch: Programming for all. *Communications of the ACM*, 52(11), 60-68. doi:10.1145/1592761.1592779

Scratch - Imagine, Program, Share. Retrieved from <https://scratch.mit.edu/>

Shabanah, S.S., et al. (2010). Designing Computer Games to Teach Algorithms. Paper presented at the 2010 Seventh International Conference on Information Technology: New Generations, Las Vegas

SilverCode. (2021-06-15) <https://www.silvercodeproject.eu/en/>

Snap! (Build Your Own Blocks) 4.2. Retrieved from <https://snap.berkeley.edu/>

Sykes, E.R. (2007). Determining the Effectiveness of the 3D Alice Programming Environment at the Computer Science I Level. *Journal of Educational Computing Research*, 36(2), 223-244. doi:10.2190/J175-Q735-1345-270M

Tashiro, J.S.D. Dunlap. (2007). The impact of realism on learning engagement in educational games. Paper presented at the Proceedings of the 2007 conference on Future Play, Toronto, Canada

Tomorrow Corporation: Human Resource Machine. Retrieved from <http://tomorrowcorporation.com/humanresourcemachine>

Vahldick, A. (2017). Aperfeiçoamento das Competências de Resolução de Problemas na Aprendizagem Intodutória de Programação de Computadores usando um jogo sério digital. Faculdade de Ciências e Tecnologia da Universidade de Coimbra, Retrieved from <http://hdl.handle.net/10316/79528>

Van Brummelen J., Phoebe Lin, (2021). Engaging Teachers to Co-Design Integrated AI Curriculum for K-12 Classrooms. ACM CHI Virtual Conference on Human Factors in Computing Systems (CHI 2021)

Vermeulen, L., et al. (2011). Girls will be girls : a study into differences in game design preferences across gender and player types. Paper presented at the Under the mask: perspectives on the gamer, Luton, UK. Retrieved from <http://hdl.handle.net/1854/LU-1886961>

Weintrop, D.U. Wilensky. (2015). To block or not to block, that is the question: students' perceptions of blocks-based programming. Paper presented at the Proceedings of the 14th International Conference on Interaction Design and Children, Boston, Massachusetts

Wolz, U., et al. (2008). 'Scratch' Your Way to Introductory CS. *SIGCSE Bulletin*, 40, 298-299.

Yu J., Using Facemesh in MIT App Inventor to Empower Students to Apply Artificial Intelligence, M.Eng thesis, Elect. Eng. Comput. Sci., Massachusetts Inst. of Technol., Cambridge, 2021

Zhu J., Creating Your Own Conversational Artificial Intelligence Agents Using Convo, a Conversational Programming System, M.Eng thesis, Elect. Eng. Comput. Sci., Massachusetts Inst. of Technol., Cambridge, 2021

Zyda, M. (2005). From Visual Simulation to Virtual Reality to Games %J Computer. 38(9), 25-32. doi:10.1109/mc.2005.297

