

Children Learning Object Oriented Programming

A Design Science Study

Bachelor of Science Thesis in Software Engineering and Management

Hampus Gunnrup
Pooriya Balavi



The Author grants to University of Gothenburg and Chalmers University of Technology the non-exclusive right to publish the Work electronically and in a non-commercial purpose make it accessible on the Internet.

The Author warrants that he/she is the author to the Work, and warrants that the Work does not contain text, pictures or other material that violates copyright law.

The Author shall, when transferring the rights of the Work to a third party (for example a publisher or a company), acknowledge the third party about this agreement. If the Author has signed a copyright agreement with a third party regarding the Work, the Author warrants hereby that he/she has obtained any necessary permission from this third party to let University of Gothenburg and Chalmers University of Technology store the Work electronically and make it accessible on the Internet.

Children Learning Object Oriented Programming

A design science study about teaching the elementary concepts of object oriented programming to children between seven and twelve years old.

Hampus Gunnrup
Pooriya Balavi

© Hampus Gunnrup, June 2017.

© Pooriya Balavi, June 2017.

Supervisor: Dave Stikkolorum

Examiner: Regina Hebig

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Cover:

The image depicts an example view of the proposed artefact, it illustrates an early stage of the game.

Children Learning Object Oriented Programming

A Design Science Study

Hampus Gunnrup
Department of Computer Science and Engineering
University of Gothenburg
Gothenburg, Sweden
gunnrup@gmail.com

Pooryia Balavi
Department of Computer Science and Engineering
University of Gothenburg
Gothenburg, Sweden
pooriya.balavi@gmail.com

Abstract—The discipline of teaching children is a topic that has been explored extensively throughout history. There are known theories about pedagogy which repeatedly have been proven to be successful, and have been used in countless cases. One area that remains unexplored, however, is teaching the concepts of object oriented programming to children. The objective of this report is to pinpoint the important characteristics of a game addressing this matter. By the use of a design science approach, the development of an artefact is presented. The artefact bases its design on known theories about children’s learning and elementary notions of object oriented programming. The evaluation of the artefact is done by the means of experiments involving seventeen students, and through analysis of observations, open-ended discussions and a paper quiz. The results indicate that it is possible to teach object oriented programming to younger children. The collected data were categorised in themes that were identified as *opinion*, *learning outcome* and *inconclusive findings*. Additionally, it is concluded that incidental learning and traditional methods of pedagogy are usable in games that teaches programming. The results will be useful both for the development of similar games, and for future research.

Keywords—Educational game, object oriented programming, OOP, children, learning, design science.

I. INTRODUCTION

There are many existing ways for learning to program. They exist for all experience levels. There are apps, toys and even board games. The trend of teaching the fundamentals of programming to children is rapidly expanding [1]. They focus on conceptualising the main concepts of programming (e.g. loops, conditions, if statements and commands). Visual programming languages teach programming logic and concepts to children even before they can read. Experiments have shown that often a complex subject of computer programming is easier to grasp at younger ages [2]. This notion can be compared to learning how to ride a bike, learning the fundamental concepts of programming is easier for children in younger ages.

However, there are very few solutions designed to specifically teach object oriented programming (OOP) to children. In this study, there are three tools that are considered as technological foundations. The first, *Scratch* [3], teaches programming to children from the age of eight years old and up. Although Scratch fails to introduce OOP, it is a highly popular and effective tool for learning programming. The second tool, *Alice* [4], is similar to Scratch, but introduces OOP. However,

Alice is targeted towards adolescents from twelve years old and up. Finally, *GreenFoot* [5], also teaches OOP and is similar to Alice. Although, GreenFoot uses actual Java code, but in a simplified manner. GreenFoot targets adolescents fourteen years old and up. As it is discussed later in the report, none of the mentioned tools can be considered as games, due to a lack of an effective feedback mechanisms, reward system and sense of achievement for the player.

Thus, one fundamental aspect of programming remains challenging, and partly unexplored, to teach children under the age of twelve years old. The concept of OOP. This mixed methods (although mostly qualitative) study, focuses on how to teach the elementary concepts of object oriented programming to children in the ages 7-12 years old, through the use of an interactive game.

A. Purpose

It is argued [6] that conceptualising objects in the real world and grasping them as components of a system is hard to grasp, not only for children but also for adults. Object-oriented systems are composed of objects that cooperate together under provided regulations to perform the required functionality [7]. It allows the use of data modeling which is the procedure of formalising a complex software system into manageable pieces of code [8]. According to Kiczales [9], learning OOP at a younger age allows a child to develop an abstract-thinking mentality. That is, to be able to comprehend the concept of designing components (objects) of a system. Furthermore, comprehending that these objects may be altered depending on different contexts, is an important concept that might be grasped. This type of thinking leads to the skill of being able to classify objects into groups, formulate their relationships, comprehend their data types and understanding that a function (method) is capable of solving a broad range of problems. According to Krajnović [10], this would potentially allow children to think in an algorithmic problem solving and object oriented manner.

As mentioned, this study explores the feasibility of teaching some concepts of OOP to children. More specifically, the study tries to assess whether children in the ages 7-12 years old can understand the concepts on an abstract level. Rather than

introducing actual programming, the design behind OOP is taught.

Through the use of a design science approach, an artefact (prototype of a game) was developed to be tested on a sample group consisting of children between the ages seven and twelve. The results were analysed and evaluated qualitatively and partly quantitatively.

B. Research Question

- Main research question:
 - RQ 1: How can the elementary concepts of object oriented programming be taught to children at early ages?
- Sub-questions
 - Sub-RQ 1.1: How do young children react to the introduction of object oriented programming concepts?
 - Sub-RQ 1.2: Is incidental learning an effective approach to teach object oriented programming to children?

II. BACKGROUND

In the domain of OOP, the metaphor of dwarfs standing on the shoulders of giants becomes highly relevant, meaning that going back to the very origin of the paradigm shows what concepts are important to teach. The languages *Simula* and *Smalltalk* can be seen as being the first adopters of the object oriented paradigm [11]. As mentioned by Kay [12], the pioneer of the concept of object orientation, the concept of classes and instances can be found as early as the 1960s in the paper about *Sketchpad* by Sutherland [13].

In this research the scope has been narrowed down to teach the concepts of classes and instances (objects), due to the origin of OOP being based on specifically that.

A. Literature Review

In this study, scientific foundations play a major role in designing the artefact and collection of qualitative data. Children's learning, object oriented programming and game design principles are the main scientific fields of study which each are explored in detail throughout the report. Furthermore, the literature regarding the technological foundations are a vital part of this research.

B. Object Oriented Programming Paradigm

According to Britton [6], in comparison with procedural programming, the OOP paradigm can be described as simply privileging data over action that is in contrast to procedural programming where a programmer approaches the problem by decomposing the system into a series of actions often refereed to as functions. Furthermore, Glasser [14] argues that the traditional way of programming is looked upon as being tedious and error prone. This can be due to an excessive number of repeated code. In OOP, a programmer approaches a problem by first trying to decompose it into subroutines of reusable code to avoid writing the same code repeatedly. This principle allows to dynamically customise the code which

potentially increases the usability, strengthens the maintenance of the code, boosts the debugging process and improves the efficiency of the designed software.

Many popular programming languages today are considered object oriented because they include features which allow or even encourage the programmer to design their code in an object oriented style. A few examples of such programming languages are; Java, C# and C++ which are statically typed and Javascript, Python and Ruby which are dynamically typed [15]. In a nutshell, OOP is a way of rationalising that *things* can be classified by their attributes and behaviours. Objects that act as building block of a software system. They usually tend to represent something from the real world. For instance, a cat, a car or a human. According to Wu [7], each object has three main characteristics or properties; identity, state and behaviour. The identity of an object records its unique ID that distinguishes it from other objects of the same class and it is used to invoke a specific object in the code. The state of an object, also known as attributes, is a representation of the object's characteristics and specification (e.g. an object's colour, age, name, etc.). Behaviour or methods of an object is defined as the actions the object can perform. While it is beyond the scope of this paper to discuss all the features of the OOP paradigm, other characteristics are; data hiding (encapsulation), message passing, composition, polymorphism and delegation.

Additionally, Britton mentions that when programmers use OOP, they start by identifying the objects of the system, and from that they are able to create classes that determine the attributes and functionality. A class is a description for objects with the same defined set of attributes, behaviours and relationships. It is a blueprint for creating actual objects. According to Glasser, an important concept of OOP is object instantiation, where an instance is created from a class. That instance gets its own separate fields compared to all other instances of the same class.

C. Children's Cognitive Science

Children's cognitive science is an extensive field of study where the learning skills and the factors that affect children's learning outcomes are analysed [16]. Jean Piaget, a psychologist who studied young children's learning abilities [17], introduced the theory of *constructionism* where he described how children are capable of comprehending and forming their own knowledge during a learning process and the knowledge does not simply convey from one person to another. In 1980, another visionary educator, mathematician and computer scientist, Papert [18], extends Piaget's theory by proposing the Piagetian learning method which simply means "learning without being taught". In this approach [19], Papert believed that children in their early ages form an understanding of the world surrounding them and they learn by experiencing the dissimilarities and contrast of what they knew previously and the new discoveries around them. Papert pioneered the use of computers as an aid to learning and argued about the positive effects of using computational technologies for edu-

cation. Therefore, it is believed that educational programming games can be used as an invaluable tool to allow children to think and solve problems in a methodical manner and help to enhance their cognitive ability; which is the process of thinking, perceiving and remembering. Additionally, an effective programming game has to encourage the necessary characteristics of cognitive development:

- 1) active engagement,
- 2) participation in groups,
- 3) frequent interaction and feedback, and
- 4) connection to real-world context.

In relation to this research, another important principle is the Bloom's hierarchy of cognition that acted as a constructive foundation on designing an effective educational game. Bloom [20], an educational psychologist, established a model that indicates the key levels of a learning process into six cognitive levels categorised by their complexity. These elements have created a solid basis for designing various types of educational tools and assessment techniques [21]. Bloom's six categories of educational objective are formed from the most basic level to the most complex and each layer is constructed based on the previous one: *knowledge, comprehension, application, analysis, synthesis, and evaluation* [22]. Following these fundamental objectives throughout the design of the game and assessment methods, provided the foundation to match the learning goals (OOP paradigm) to the game-play (intellectual puzzle).

III. MOTIVATION

A. Existing Solutions and Problem Domain

There are many resources to learn programming from. Various educational games in different platforms focus on providing this service to children, for instance; *Scratch*, *Kodable*, *CodeCombat*, *Box Island*, *Bit By Bit* and *Lightbot Hour* are a few examples of such educational applications. However they mainly concentrate on the fundamental idea of programming which is planning, testing, debugging, procedures, sequences, loops and the overall logic of programming.

After a series of investigations, it was concluded that there are only three educational tools that can be used as a technological foundation; *Scratch* since it solves the issue of teaching real programming to children eight to twelve years old, *GreenFoot* since it teaches OOP and *Alice* that does the same as GreenFoot.

The first solution, *Scratch*, solves the issue of teaching imperative programming to children. This is done by the use of colorful visual command blocks that control a 2D world. The 2D world is composed of a background called the *stage* and graphical objects called *sprites*. By allowing the user to change the background, create new sprites, and manipulate properties regarding position and presentation, they are in fact experiencing programming while having fun. *Scratch* uses a minimalistic interface and aims to be self explanatory. There are no specific goals, hence it is not a game, but rather an

open environment for children of all ages to experiment and understand the basics of programming. *Scratch* is not object oriented (although there is an extensions which is discussed later in the report, that brings object oriented concepts to *Scratch*) and barely introduces procedures as a concept. It focuses more on what Resnick and Rosenbaum [23] define as *tinkerability*, "a playful, experimental, iterative style of engagement, in which makers are continually reassessing their goals, exploring new paths, and imagining new possibilities".

GreenFoot, another existing solution, uses actual code but simplifies the learning by eliminating the need to initially write any. Users can easily create a scenario with minimal functionality, yet still experience the concept of object orientation. They can also experiment with existing scenarios, and learn in a scaffolding manner. *GreenFoot* focuses on students from around fourteen years old and, like *Scratch*, uses a two dimensional environment that the students can interact with. *GreenFoot* does not aim to create a gentle introduction for younger children. Rather, it intends to remove the less important administrative tasks, such as the need of a main method, and puts the focus on OOP.

The final existing solution, *Alice 3.3*, is an open-source game that is targeted towards middle school and up, as well as university students who want to learn OOP fundamentals through 3D animations. The game-play is designed in a way that resembles the real world and how students interact with the objects around them. The game's interactive IDE allows drag-and-dropping various objects into a scenario and assigning properties and commands to each of them. The player can run the animation they created and observe the result by visualising the scenario. According to a research [24] done on *Alice*, the tool's incremental construction approach, aims to not only illustrate the concepts of objects and classes in an OOP manner but also contextualise the required pseudo-code to develop designs via open-ended assignments [25].

B. Gap in the Existing Solutions

All of the mentioned technological foundations use an intuitive visual approach. However, *Scratch*, which addresses a younger audience (from eight years old) compared to the other two technologies, lack the possibility of introducing OOP concepts. One might argue that *Scratch* contains OOP since the extension, *Snap!* [26], introduces it. While this is true, it removes one of the key features of *Scratch*, its accessibility towards younger children. According to Harvey et al., *Snap!* targets ages of fourteen and above which overlaps with *Alice* and *GreenFoot*.

GreenFoot tries to solve the problem of teaching OOP in a more concise and comprehensible manner, but addresses older teens instead of younger children. It is commonly used in high school or college courses, or even introductory university courses.

Even though *Alice*'s interactive environment makes it one of the best available tools for learning OOP it still has a few noticeable drawbacks. Complex style nature of *Alice* may not be suitable for all students as not everyone has the same set

of knowledge and abilities. Alice 3.3 encourages the learner to solve problems by exploring a simulated environment, however this may bore the learner as there are no tasks to achieve and the reward system is almost non existing which may lead to a shorter attention span. Furthermore, despite its unique context for understanding OOP paradigms, it cannot be classified as a game. Alice's lack of feedback, a reward system and a sense of accomplishment diminishes its relation to the category of educational games [27]. This could potentially have a negative impact on the pedagogical contribution of the game.

All three of these examples bring their part to the world of teaching programming to beginners of all ages. Although, there is a noticeable gap. Namely the problem of teaching OOP to young children (younger than 12 years old). The improvement would be to use the learning goals and outcomes of the existing solutions that teaches OOP, but in a simplified manner that can be understood by a previously unexplored target group; children in the ages six to twelve. Moreover, the existing solutions all present an environment where students can play around and become familiar with the concepts, rather than solving specific tasks and getting feedback. Therefore, a further improvement would be to implement tasks that the user can complete and give rich feedback that clearly shows what does and does not work in regards to teaching OOP to children.

C. Characteristics of an Educational Game

Every individual has a different level of skills when it comes to reading, thinking and absorbing information. It is argued [16] that using a game as a learning tool does not require the same level of cognitive skills. This is due to the different experiences that each learner faces by playing the game as well as the diverse learning outcomes that it brings. Educational games have become a powerful way of learning. According to numerous researchers [27], an effective educational game is capable of encouraging learners to solve tasks through the exploration of simulated environments. While it is argued [28] that having fun when playing an educational game can lead to better learning outcomes, the characteristics of such games strive for a balance between the learning objectives and enjoyment. Furthermore, a competent educational game should provide challenges, support and feedback throughout the game. The presence of a reward system in the game for completion of tasks can significantly increase the competitiveness of the game and give a sense of motivation as the learners will try to beat the game or the previous high scores. Hence implementing a challenging reward mechanism is an essential feature of any game [22]. Another important aspect of educational games is the concept of *incidental* (indirect) *learning*. In this context, incidental learning is described as when young learners do not necessarily know they are obtaining knowledge by playing a game [27].

Gee, a psycholinguistics educator, argues [16] the importance of *connectionism* (i.e. discovering patterns in our experiences) in educational games. He strongly stresses that human

beings are great at recognising patterns and by following this principle, even challenging concepts can be taught to learners. According to Gee's research and personal experience, people often do not contemplate best when they try to reason via theoretical (general abstract principles) learning and on the other hand contemplate best when they reason via experimental learning (patterns they picked up from actual experiences).

In this research, the development of the game design documentation (GDD) began at the early stages of the study and its content gradually increased. GDD [28] is a scheme for designing any type of game. Through the use of this documentation technique, important aspects of the game were planned and stored before developing the artefact. Aspects such as; the game-play description, the game's purpose and its educational outcomes, the demographic (targeted group of users), its software architecture, the requirements, use-case scenarios and UML diagrams, audio requirements (voice recordings and its analysis) and the overview of all levels of the game.

Through the use of *situated cognition* and the adaptation of connectionism theory and incidental learning, the elementary concepts of OOP were illustrated in the game. By visualising where young learners can understand and build the mental model of the OOP paradigm [29]. Additionally, the designed game is comprised of an interactive feedback mechanism, fun game-play and a basic reward system which categorises the game in the group of educational games.

D. Proposed Solution and its Disadvantage and Advantages

Despite three existing tools, there is still a notable lack of resources on teaching OOP to children. One possible factor could be the difficulty of the subject and the lack of research within the field. The researchers of this study aimed to fill this gap by attempting to create a prototype of a game that teaches the elementary concepts of OOP. Although, it is important to consider all advantages and drawbacks of the research. They are defined as follows.

Advantages:

- The proposed game is considerably simpler to interact with. It follows storytelling and task-based scenarios to visualise the difficult notion of OOP to young children. Compared with other solutions that use an open environment for the students to play around with and learn the concepts from.
- The game's interaction with the children will be through visualisation of objects and classes in a fun and interactive manner. Additionally, the game largely relies on a voice that narrates a story, defines the tasks for the players and motivates them when they accomplish tasks.
- The existing solutions often require a supervisor/teacher that aids the children in the process. Our solution may not require an adult to the fullest extent as the voice guides the child throughout the game.
- Through the use of the voice, a feedback mechanism was implemented, notifying the users who do not follow the defined tasks or click on the wrong section of the page.

- The game may increase female interest in computer science, which ultimately improves the gender quotation within the field.
- Such an educational game is capable of invoking children’s critical and computational thinking, which improves the overall cognitive abilities of children and shape the essential way of reasoning and problem-solving required to learn how to program [30].
- By introducing the concept of OOP to children, they are familiarised with a widely-used programming method. Compared to procedural programming, OOP is highly valued in the industry [6] as it allows a stronger maintenance, ease of documenting code, fastening the debugging process and data hiding.
- The final game can be used as a tool in primary schools’ curriculum to conceptualise the paradigms of OOP.
- The conducted research assesses the effectiveness of incidental learning on children, specifically in the realm of OOP. This improves the knowledge base, and can be used in future research and solutions.

Drawbacks:

- The proposed solution will not teach actual programming, as other existing solutions do (GreenFoot and Alice). However, this is the purpose of the game; to teach the concepts of OOP in an indirect manner and focus on helping children to think in an object oriented way of thinking. Introducing real programming would be the next step, ideally by suggesting Alice.
- The existing solutions teach more advanced features of OOP such as polymorphism, public interfaces, delegation, message passing, data hiding and inheritance.
- Some may argue that promoting such technological tools to children may have negative effects, for example; encouraging the lack of group work.
- Due to a lack of resources for the development, the game will most likely be in the form of a prototype. During later stages, through the use of findings, the artefact can be improved.

IV. METHODOLOGY

A. Definitions

This section is dedicated to the definition of the mentioned concepts in the report. This is needed, since some concepts that are used repeatedly needs to be clarified. Figure 1 shows a table of these definitions with a brief description for each.

Furthermore, this study attempts to assess the feasibility of creating a game for children that helps them comprehend the elementary concepts of OOP, specifically understanding:

- Objects and classes (a class is referred in the study as a *group*),
- Classification and instantiating of objects,
- Attributes or behaviour of a class and related objects (referred in the research as *description*),
- Modifying the value an attribute, and
- Adding an attribute to an object or class.

OOP	Object oriented programming
Game	Referring to anything that is close to a game. Some of the mentioned solutions, cannot be defined as games, but in a few occasions we may refer to them as such
Artefact	It is the designed game, used as a medium to teach the elementary concepts of OOP to children
Participant	Generally, refers to the samples or the children from the age of 7-12 years old, who took part in testing the artefact
Paper quiz	It addresses the objective test that was designed to gauge the introduced concepts of OOP from participants. This test was carried out in the form of traditional pen-and-paper

Fig. 1. Definitions of the mentioned concepts in the report

B. Research Design

This study uses a design science [31] approach together with a mixed methods design [32]. Adopting a mixed methods design provides the opportunity for using both qualitative and quantitative data. Figure 2 indicates the main stages of the research, in a chronological order starting from the top left and ending in the bottom left. Furthermore, an important phase that is not depicted in the figure is the literature review of the related topics. This was due to the continuous research that was carried out throughout the whole study.

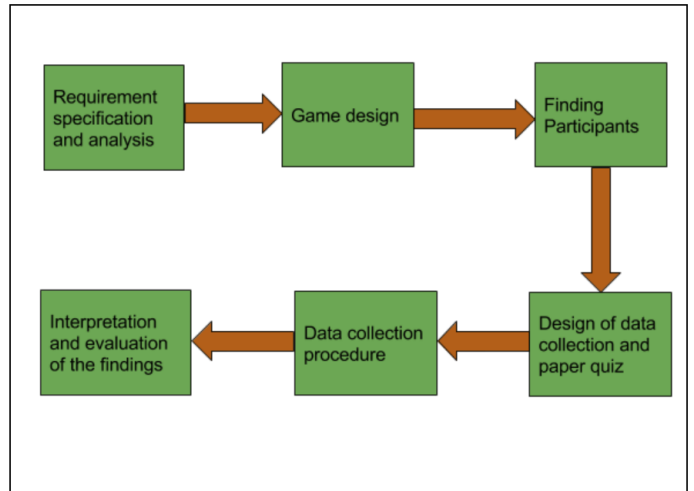


Fig. 2. The overall framework of the research

Mixed methods(also called *integrating*, *synthesis* and *multimethod*) was chosen in this study because of the strong results that are gained when combining both qualitative and quantitative analysis. Furthermore, using mixed methods allows for a better understanding of experimental results, by comparing the different perspectives from the quantitative and qualitative data. However, during late phases of the study, the researchers discovered that not much quantitative data was retrieved. Thus, the main quantitative results is in the form of statistics regarding the perception of the participants (i.e. if the proposed artefact was fun to play, if it was hard to use and if the participants learned anything). These statistical results were later compared with the analysis of

the transcribed observations, interview data and qualitatively analysed answers of the quizzes.

The type of mixed methods design in this study is defined as *convergent parallel*, which means that the qualitative and quantitative phases occur concurrently, or in no specific order. This choice is due to the innovative approach of the research. The required results are proof of if and how OOP can be taught to young children, and the data is collected during closely examined sessions with each participant, thus the data is strongest when it is triangulated. For this, a convergent parallel approach is ideal. The other two main options are *explanatory sequential* and *exploratory sequential*, which means that either the quantitative phase comes first or the qualitative phase comes first, respectively. Since these approaches typically require a stronger background and more time, they were disregarded in this research.

There are, however, some downsides of using a mixed approach. The need for both collecting and analysing the two types of data is far more time consuming than only using one type of data. In this research, as mentioned, we limit the amount of quantitative data, as the innovative nature of the topic does not fit the needs of the quantitative approach. Furthermore, according to Shaw [33] the chosen samples (children seven to twelve years old) are best studied in an open and informal atmosphere, and children under twelve are not fit to answer a self-completion questionnaire. Although, this does not mean that quantitative methods cannot be used, qualitative methods are a better fit.

The main question of the study is about feasibility - whether it is possible to teach the notion of OOP to children and if so, how can it be achieved? Therefore, the chosen strategy falls under the category of constructing a system. Following the construction, a validation phase was conducted in order to evaluate the feasibility of the topic and argue about the results of the study.

Following the evaluation methods proposed by Hevner [31], this research uses a mix of *observational*, *experimental* and *descriptive* evaluation by conducting a scheme for each participant (i.e. testing the knowledge of the participant before and after using the artefact) that is similar to controlled experiments and conducting open-ended interviews. Additionally, observations were transcribed during every step of the collection.

C. Artefact

The development of the prototype followed what it aims to teach, namely OOP. The prototype is built in a language (Javascript) that supports object oriented design. The requirements and design of the prototype is documented by the use of UML. A partial component diagram of the artefact can be seen in Figure 3. This shows an overview of the main components of the artefact.

- 1) The *Game* component acts as a controller for the main core of the game, and provides an interface for handling input/output, graphics, touch events (or click events) and the dimensions of the game. Furthermore, the *Game*

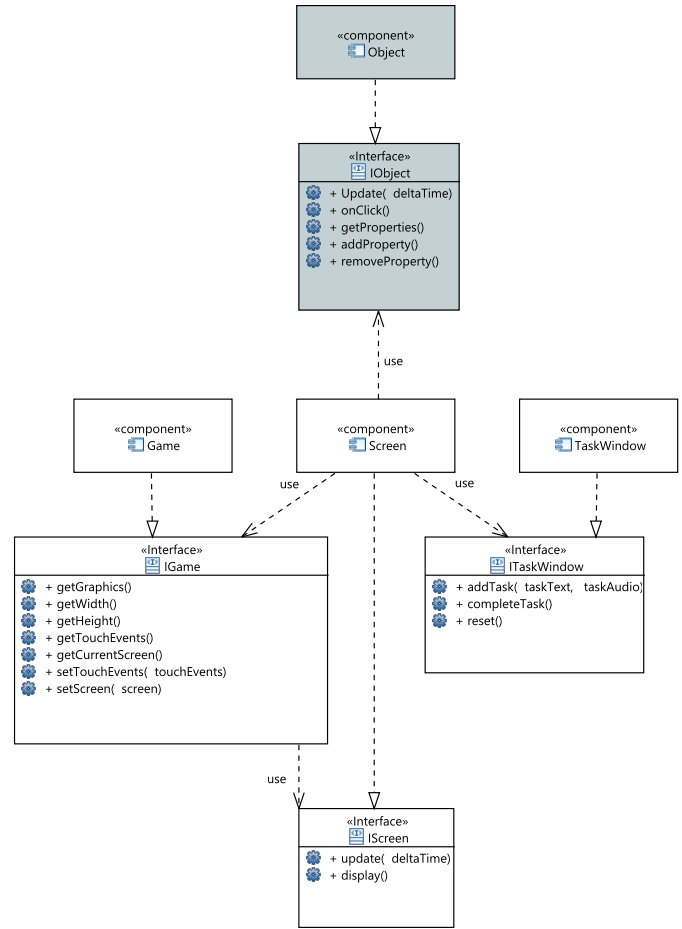


Fig. 3. Component diagram of the artefact

- 2) The *TaskWindow* component handles all tasks throughout the game. It provides three main operations in its interface, for adding tasks completing tasks and resetting the whole *TaskWindow* (removes all of the current tasks). When all of the tasks are completed, the *TaskWindow* component presents a continue option to the user. This starts the next scenario (or goes back to the menu screen).
- 3) The *Object* component represents everything on the screen that is not part of the *TaskWindow* or part of the background. Most elements handled by the *Object* component is clickable, which results in a window appearing next to the element showing its properties (see the owl in Figure 6).
- 4) Finally, The *Screen* component is responsible for putting everything together. Every scenario of the game (and the main menu) uses its own *Screen*. The *Screen* component uses the *Game*, *TaskWindow* and *Object* components in order to build each scenario of the artefact.

The artefact of this research is, as mentioned, a prototype

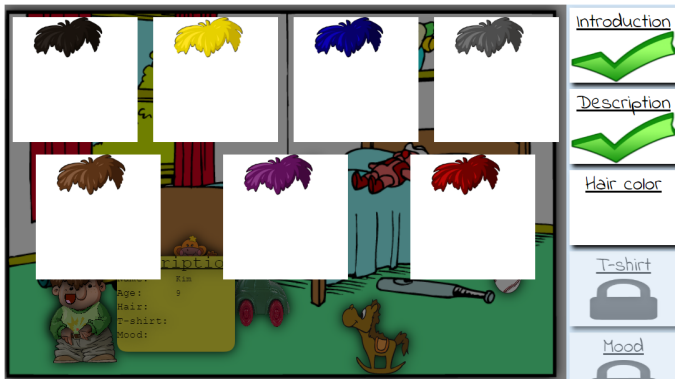


Fig. 4. The first scenario of the artefact (objects and properties)

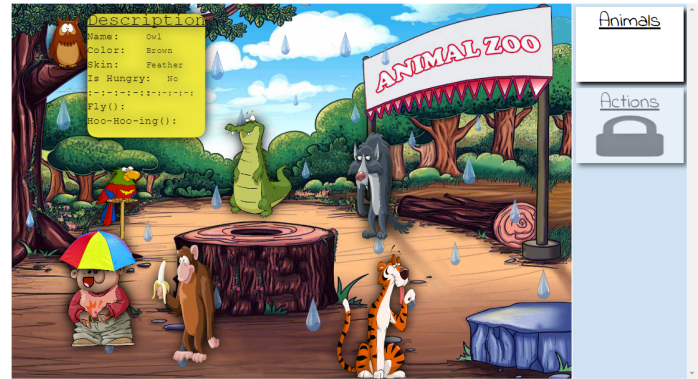


Fig. 6. The third scenario of the artefact (objects from the same class and introduction to methods)



Fig. 5. The second scenario of the artefact (class and its properties)

of them have a specific set of attributes (called description). This is described in a way that supports incidental learning, by saying "let's call these objects from now on" rather than stating that "these are called objects". This style of incidental learning is used throughout all of the tasks in the game. The second scenario introduces classes. The same principles of teaching is applied as in the first scenario. The main task involves adding a new property (umbrella hat) to the class of human (see Figure 5), and viewing the related objects of that class, in order to see that the property is added. This scenario tries to show that all of the objects on the screen have a few things in common and can be derived from the same class. By allowing the child to add a property to the shared class, and then visually showing that the property is added to all of the instances of that class, the child could understand how classes and objects are connected in an abstract level.

of a game. It aims to show how the elementary concepts of OOP could be taught to children. It uses the theories about pedagogical learning, game design and OOP that is mentioned in this report. The game introduces tasks that are explained through a voice. When the voice is active the main character (Kim) in the game displays a *speech bubble*, thus depicting that Kim is speaking. Furthermore, the voice is implemented to act as a feedback mechanism, alerting the users that do not follow the provided instructions. It also encouraged the users after completing specific tasks. The voice praises a users performance by saying "good job!", "nice choice of colour!" or "woow, you have done a great job so far!". In addition to this, the users were rewarded by receiving stars after completing a scenario in the game, which gave a sense of accomplishment.

In the third and final scenario, the user is presented with a zoo setting (see Figure 6), where there are different types of animals that are clickable. In this scenario methods (actions) are introduced. Compared to the other scenarios, the user does not perform many tasks. The main learning outcome comes from the voice that explains the concept of an object having a set of actions (methods) and the difference between objects of two different classes (classes of human and animals).

Appendix A shows a complete sequence of the game.

D. Data Collection

The samples that were used in this study are put in two categories. Firstly, the largest group of samples were twelve students aged between eleven and twelve years old. All of the students are from a school near Gothenburg, Sweden. The researchers spent one full day collecting data using the same process for each participant. Each participant conducted the experiment separately, and the process took around 15-20 minutes. The second group of samples are five students from an international non-profit organisation called CoderDojo, that teaches programming to children. These children were in the ages seven to twelve. As with the school, the children conducted the experiment separately. In both situations, a random sampling approach was carried out in order to find participants.

The samples were randomly selected by the teachers of the school based on a specific requested age range.

For each of the seventeen participants, the environment was quiet and relaxed and the researchers aided the participant when needed without affecting participant's answers. Each child was assured that there were no wrong answers. Finally, the whole process was closely monitored, each discussion session was recorded and every detail that the researchers found to be worth noting was transcribed.

As the name of the mixed methods approach in this study suggests, the collection of data was conducted in parallel. The qualitative data was collected using *open-ended interviews*, *observations* and a *paper quiz*. The quantitative data was collected from a small subset of the questions in the paper quiz. As Creswell [32] mentions, the main idea of using a this type of mixed approach is to collect both types of data using the same variables and concepts. This is achieved by using the same quiz for both qualitative and quantitative data.

Measuring the learning outcomes of the artefact and evaluating whether the involved participants understand the concepts of OOP is a decisive aspect of this research. Although both quantitative and qualitative data was gathered, the evaluation largely relies on qualitative data [32]. Primarily, informal open-ended interviews were used in order to provide a broad view on the children's understanding of the introduced concepts. It allowed the researchers to directly ask questions such as; "What did you learn in the game?" or ask about a specific introduced concept, for instance "How would you describe an object or properties of an object?" or "Was the game fun? Would you tell your friends about the game?" In addition to this, observing participants, their reaction and body language while playing the game was closely monitored and recorded. Figure 7 visually presents the main stages of the data collection.

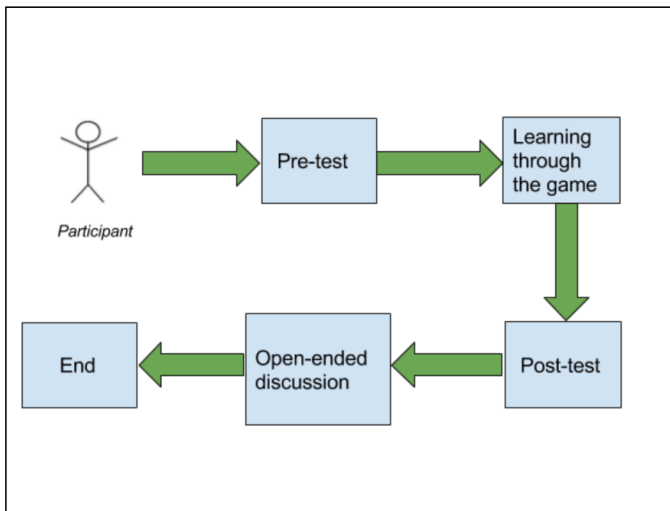


Fig. 7. Main phases of data collection

As mentioned, a paper quiz was designed to test the OOP knowledge of the participants. This test is used in the form

of traditional pen-and-paper style to evaluate the participant's understanding of the elementary concepts of OOP. When designing the paper quiz, Bloom's taxonomy of educational objectives [22] was investigated in order to increase the quality of the test and to be able to adequately measure the knowledge of the participants. The learning objective of the paper quiz is summarised below, according to Bloom's six principles:

- 1) Knowledge: being able to recall a previously learnt knowledge.
- 2) Comprehension: understanding or translation of the perceived information in one's own words.
- 3) Application: the ability to implement the learnt theories and concepts in new situations.
- 4) Analysis: ability to break down and identify patterns.
- 5) Synthesis: generalisation of the perceived knowledge and be able to use the information to build new ones.
- 6) Evaluation: refers to the ability of comparison and discrimination between ideas.

The paper quiz follows the specifications of an objective test [34]. This type of assessment is often used for evaluating specific aspects of students learning and is considered to be an effective tool for examining their *recall of principles*, *knowledge* and *practice of terms*. The questions in the designed quiz mostly consist of selecting the solution from a set of options (multiple choice and matching choices) and brief answering via the use of inputted text or number. Furthermore, the paper quiz was used to conduct a *think aloud* method [35]. For example; the researchers pointed at a specific question in the paper quiz and asked the participant to reason about the principles they followed to solve that question, thus concluding if they have used the concepts of OOP to solve the question or not.

The paper quiz consist of two phases; a pre-quiz which was carried out before a participant played the game and a post-quiz that was conducted after playing the game. The questions designed in both quizzes had the same structure but the content of each question was different. The learning objectives of each question in both paper quizzes and their relation to the concepts of OOP is discussed and rationalised below:

1) *The Pre-test*: This paper quiz was designed to assess the participants knowledge of OOP before playing the game. The quiz consists of three questions (see Appendix B).

In the first question, the concepts of groups (classes) were tested and later in the open ended discussion, each participant were asked to reason about why they chose the selected groups. In this question, the participant looked at the similarities of the images/characters and categorised them according to their appearance. This can be related to the definition of a class in OOP terminology.

In the second question the notion of properties is illustrated through a box that represents the *description of an object*. This description box is similar to a class in a UML class diagram. The goal of this question is to test the participants' previous understanding of what properties are. This helps the evaluation of the similar questions in the post-quiz, by comparing the answers.

Odd (complex) results	Somewhat correct
Somewhat fun	Incorrect answer
Fun	Inconclusive
Not fun	Not incidental
Improved answer	Incidental learning

Fig. 8. The codes and colours used in the analysis

In the last question of the paper quiz, the participant is required to name the group of animals and circle the image that could belong to that group. The concept of objects from the same class are presented. As with the previous question, the answers was compared with the post-quiz answers, which allows the researchers to assess if the a participant has learnt anything.

2) *The Post-test:* In this section the six questions from the post-quiz are discussed. This quiz has three additional questions (see Appendix C) compared to the pre-quiz due to the fact that the participant had to play the game to be able to answer some specific questions, for example object instantiating or defining attributes of a class.

In the first question, similar to the pre-quiz, the participant are asked to classify objects into different classes.

In the second question, the participants are required to describe each image (object), in other words assign values for each attribute of an object. Additionally, in the last part of the question, the participant is required to add a new attribute for the image of a little girl. This answer depicts if the participant has understood the relation between objects and attributes.

The third question is related to objects from the same class and the fact that each object share the same attributes only with different values. Hence the participant's answer could be different depending on their creativity. The most expected answers are: colour, the car's model, plate number, number of passengers and size. In addition to this question, the fourth question asks the participant to circle the object that belongs to the previous group (group of cars). The objective is to see if the children can identify a group in a manner consistent with the design of OOP.

Question 5 intended to evaluate how the participants differentiate the images. In other words, whether they can see the images as two objects that are from the same group and share similar attributes, but with different values.

In the last question of the post-quiz, the concept of instantiating objects from a class is tested. This task is designed to assess whether a participant can understand the notion of creating an object from a class. The participant is required to use its imagination to instantiate a car and assign values to its attributes. The goal of this question is to clarify if the participants can see the relation between classes and objects.

E. Analysis

The collected data was mostly analysed qualitatively. During the collection phase, the researchers wrote down early conclusions about a participant if it appeared. These early conclusions were saved for future reference. An example of such a conclusion is: *a participant grasped the concept of objects*. These transcriptions alone does not justify the final results of this research, but can be used when triangulating the final conclusions.

The main part of the data analysis followed the structure proposed by Creswell, where the data was organised and coded. The codes were later collected and themes were defined and labelled. Finally, the themes were compared and interrelated and the meaning of the results was discussed. Each step was conducted as follows:

- 1) **Organising and reading through the data:** By putting all of the collected data into one single file, an overview of the gathered information was be gained. During the process the parts of the collected data that was considered irrelevant was disregarded and most of the data was summarised, thus leading to a more comprehensible collection. As a final step, the organised data was read thoroughly, and early conclusions were transcribed by each of the researchers.
- 2) **Coding the data:** The sorted data was coded, through the use of colours (see Figure 8). Each colour was assigned a label. For instance, one code, *Fun* was assigned to any data that suggested that a participant enjoyed the game. Another, *Improved answer*, was assigned to any answer that was seen as an improvement in the post-quiz compared to the answer in the pre-quiz.
- 3) **Theming the codes:** Three major themes were defined in this study. *Opinion*, *Learning outcome* and *Inconclusive*. Any result that explains an opinion of a participant (e.g. the game was fun or the game was not fun) was put under the opinion label. Any result concerning the final learning outcome of a participant was put under the learning outcome label. This may be a finding that suggests a participant has not learned anything (incorrect answer), or that the participant has learned something (somewhat correct) or that the participant has learned exactly what the game strives to teach (improved answer). Lastly, any code that regarded data that was inconclusive or odd, was put under the inconclusive theme.
- 4) **Interpreting the results:** The final step of the analysis was to interpret the results using the defined codes and themes. The researchers individually evaluated the learning outcomes of the concepts *classes*, *objects* and *properties* for each sample. Each participant was given either pass, fail or partial pass for each of the three concepts. Following the individual evaluation, the researchers discussed the reasoning behind the grade in order to come to a final conclusion. The results that were coded as inconclusive resulted in a fail. Where

the researchers agreed, that grade automatically became the final conclusion. Finally, wherever the researchers disagreed, a thorough discussion regarding the final grade was held.

Furthermore, the relevant literature (i.e. books, articles, and similar research papers) are compared against the findings, in order to define conclusions about the results.

V. RESULTS

A total of 17 participants were examined, 4 were female and 13 were male. Figure 9 shows participants' range of age and the male to female ratio. Twelve out of seventeen participants (70 percent) had experience of playing other programming games such as Hour of code, CodeCombat and Scratch.

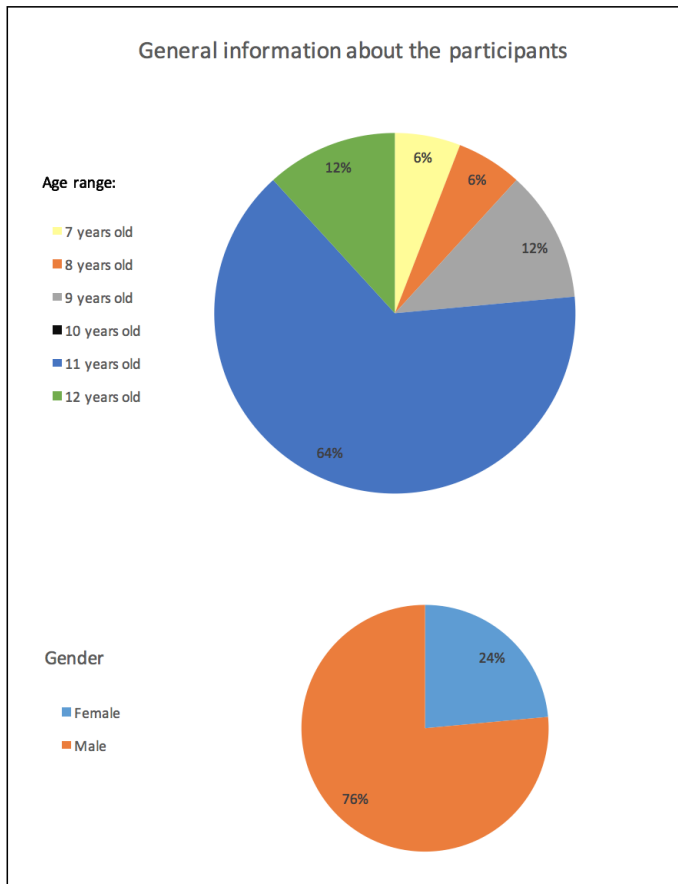


Fig. 9. General information about the participants - Distribution of age and gender

Figure 10 represents the children's perception of the game (i.e. if it was fun or not and whether they had any additional comments). Here it can be seen that a large percentage of the children enjoyed something about the game. Figure 11 presents the precise answers of the interviews when the participants were asked "Did you have fun when you played the game? What did you like or dislike about it?". 47 percent of the samples (eight participants), claimed they enjoyed playing the game, while four of the participants did not enjoy and five were partly entertained by the game. The result shows that all

the participants below the age of 10 enjoyed playing the game.

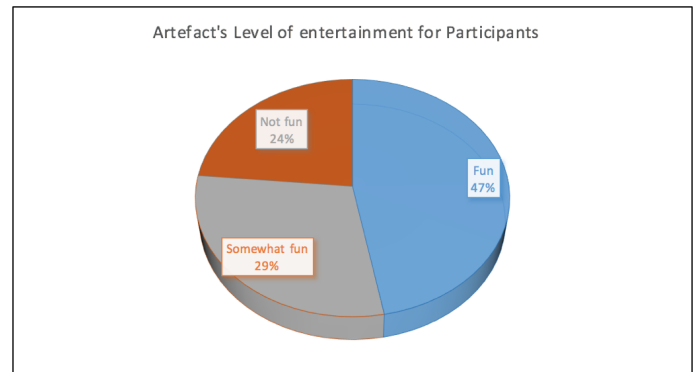


Fig. 10. Level of entertainment for participants while playing the game

Participant Number	Age	Participant's answer to the question; "Did you have fun when you played the game?"
1	11	"Sort of! Fun to learn. You can play it from time to time and keep learning. Maybe it should have had a higher level of difficulty"
2	11	"Yes! Because you get to choose the hair, shirt and mood of a character and follow the instructions."
3	11	"A little! It felt like it's for younger kids and was kinda of easy"
4	12	"No! it wasn't very fun! You just click around and little boring. I wanted to have more freedom in the game."
5	11	"Yeah! It wasn't the most fun game I have ever played, but it was still good. The story should have moved on a bit faster. The voices took too long to explain the tasks."
6	11	"It was a little fun!"
7	11	" It was a fun game. It was fun to listen to the instructions and see what you did wrong and then correct your mistakes."
8	11	"No, it was kinda of boring actually. Nothing happened. All you did was listening to a girl explaining things."
9	11	"sort of! It works! You learned things through the game. Nothing that I did not like."
10	11	"Sort of. It seemed to be for younger children. It was very linear. It was good that you learned something."
11	11	"No! I would not go home and play it. The character was funny and little weird."
12	11	"It was pretty fun. It should be a bit longer and I wanted do more stuff in the Zoo!"
13	7	"Yes, it was fun. Add more levels!"
14	12	"Yes! It was fun to fix the character. I want the game to be longer and I want to be able to do more things."
15	9	"Yes!"
16	9	"Yes! It was fun to see the properties."
17	8	"Yes, it was fun to decide the properties like the haircolor and mood."

Fig. 11. Table of participants comments

As figure 12 depicts, 82 percent of the participants indicated that they did not know what they were being taught. These results are related to the participants answers in the open-ended interviews to the question "Do you understand what we are trying to teach in the game?" and "Did you learn anything from the game?". When the researchers asked these questions,

14 of the participants either expressed that they did not know what they had learnt, or gave an answer that did not follow what they had learnt. This indicates the presence of incidental learning. For instance, a few participants said; *"the goal of the game was to follow instructions"*, another suggested *"I learned that there are more things to programming than to only make them move. That you can make pretty complicated stuff but still simple"*. Only a few participants comprehended the learning objective of the game. For instance, one participant explained *"The game tried to show that things or objects can come from the same group. And they all have different properties"*.

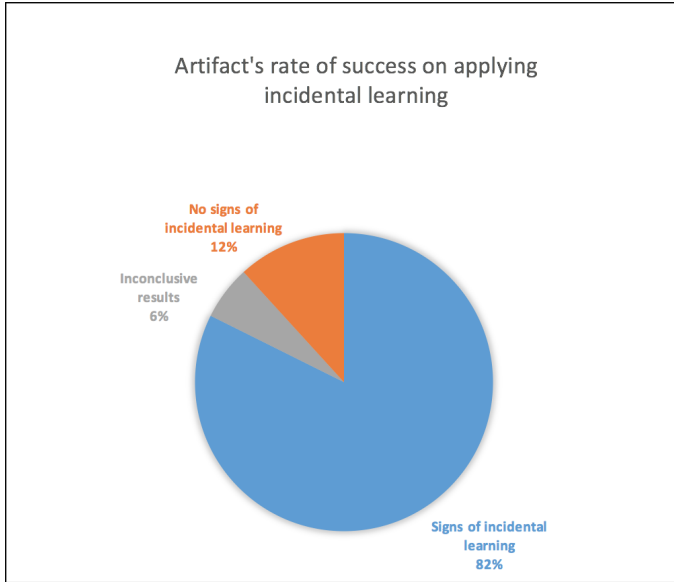


Fig. 12. Application of incidental learning in the game

Figure 13 demonstrates the researchers' final evaluation of each participant's level of knowledge regarding the introduced OOP concepts. The results of the paper quizzes, observations and most importantly the open-ended interviews, were the major data used in the evaluation. Additionally, the participants' explanations when describing the concepts of objects, classes and properties provided the basis for assessing whether a participant can understand the concepts at an abstract level, and think in an OOP manner. Specifically, comprehending objects as instances that can be created from a class that share the same properties, rather than describing an object as a tangible thing. Some participants used exemplification to describe these concepts while others attempted to describe the terminology of each concept. Following are a set of quotes from significant participants when describing the concepts of objects, classes and properties (the used names are fictional):

- Matilda, 12 years old: *"A group is something that has something in common, for example trees. Properties of a group might be size, colour or shape of the leaves. Objects of the same group might share colour, size and purpose. If tree is a group, an object is a pine tree"*. Matilda was chosen, since the final conclusion was that

she learnt everything that the game aimed to teach. She shows a typical example of what the researchers constituted as a correctly answered quiz, as she expressed herself adequately through the use of exemplification.

- Alex, 11 years old: *"An object is part of a group. The objects fish, crab and starfish are from the group fishes. Properties are things that an object is or can do. I learned that things have different properties"*. It was shown that this participant was not confident enough when describing the concepts of objects, classes and properties. His answers in the paper quizzes and open-ended interview showed that he had understood the concepts of classes but only partly comprehended objects and properties.
- Peter, 11 years old: *"Properties can be everything from being able to write with it or eat with it. A group is many things that are just a little different but still have most things in common. For example, animals and humans. Animals are in the wild and humans are in schools. An object is everything that you can touch!"*. This participant showed correct results in the quizzes, though when he tried explaining the concepts in the open-ended interview, he appeared to think of objects as tangle things and could not relate to objects of the same class. Hence his knowledge of properties was assumed to be somewhat sufficient (partly understood) but he did not grasp objects and classes in an OOP manner.

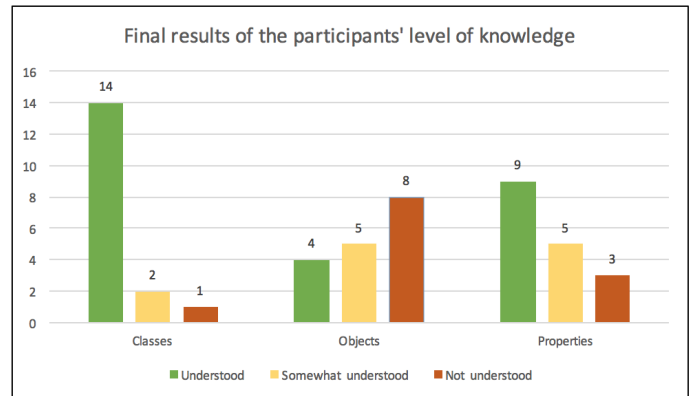


Fig. 13. Participants' level of understanding about the concepts of OOP

A. Discussion

Studying children is always a difficult process. This becomes increasingly true when the topic is previously unexplored, as in this study. Therefore approaching the data collection from a qualitative perspective, assures that no important details are left out. Furthermore, creating collection tools, and in this case an artefact, that is meant to be used by children puts extra emphasis on the language and presentation. This was solved by studying the mentioned technological foundations and relevant literature and thoroughly discussing the tools with our supervisor. The face-to-face interviews with the participants after they played the game allowed the researchers to ask direct questions to the children about how they would describe

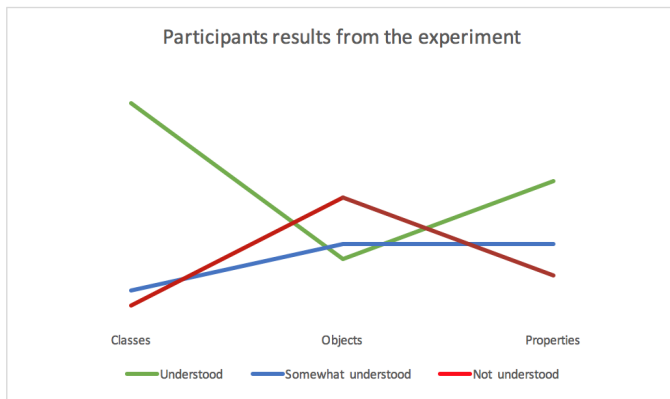


Fig. 14. Interpretation of the participants knowledge about the introduced concepts of OOP

certain OOP concepts or how they answered a specific question in the quiz. In the interviews, we expected the participants to give answers that would not bring sufficient results, that would allow us to triangulate the data with the results of the quiz. However, the answers of these open-ended questions proved to be the most valuable results, and allowed us to come to much clearer conclusions. Additionally, paying close attention to each participant's body language while playing the game helped us assume whether they enjoyed playing the game.

Figure 14 indicates the final results of the experiment. It becomes clear that the concept of classes (groups) was the most understood notion among the participants, where 82 percent of them proved to be able to think of classes in an OOP manner, whereas the concept of properties were only understood by 52 percent of the samples. The concepts of objects was presumed to be the most easy notion for the participants. But on the contrary this showed to be the most difficult concept to understand, where 23 percent showed signs of comprehending objects in an OOP manner. This is most likely due to the fact that it was not showed well enough in the game (a possible improvement would be to add tasks where the user has the chance to instantiate an object from a class). Therefore it became rather challenging for the researchers to assess the participants knowledge about objects. The main evaluation of the learning outcomes regarding objects, came from the question "How would you describe an object?" in the open-ended interview. more than half of the children (52 percent) seemed to consider an object as a tangible thing. This resulted in a conclusion that they have not learned the concept.

The age gap in this study derives from the fact that no OOP learning existed in that age range. However, this wide gap affected the results greatly. Especially the results regarding the entertainment level of the artefact. Although, most of the older participants expressed that the game was at least somewhat fun, many of them also suggested that it seemed more fit for younger children or that it was simply not fun. On the contrary, all of the children under the age of ten, experienced the game to be fun. This has to be taken into consideration when evaluating the quality of the proposed artefact - it was

fun even for most of the older children.

An interesting observation is that a lot of the participants answers in the quizzes were similar. For instance, on the first question of the second quiz, most children answered *flowers, trees* and *fruits*. Additionally, almost all of them seemed to get stuck, even after the researchers assured them that there are no wrong answers. This was especially apparent in the second question and in the last question of the post-quiz, where the participants had to use their creativity. Furthermore, most of the participants did not understand the concept of filling in the name of the girl in the second question. This is a good example of the limitations of working with children.

An important achievement that this study brought was applicability of incidental learning to teach the concepts of OOP to younger children. The researchers applied this theory when designing the artefact and assumed that the participants will not be able to understand the objective of the game and rather think of it as just playing a fun game. When the participants were asked what they thought they were being taught in the game, approximately 82 percent of them said "I don't know". This results clearly shows the presence of incidental learning and represent that it is possible to teach the concepts of OOP to children through this technique.

The worst kind of results are the ones that are inconclusive. In this research there were a few occurrences of inconclusive results. Firstly, we noticed that the last question of the post-quiz did not give any valuable results. Even the answers that were correct according to us, did not prove that the concept of instantiating an object had been learnt. Additionally, there were a few cases where the participants answered the questions consistently between the two quizzes, meaning that they seemed to answer with the same reasoning. In some cases the children even answered correctly (according to us) in the first quiz but not in the post-quiz. These type of situations resulted in a inconclusive code during the analysis phase, since we could not prove that anything had or had not been taught.

One interesting point that has not yet been discussed in this report, is how the researchers evaluated the results. In other words, what constitutes as a correct answer? As discussed in the analysis section, the data that indicated any type of improved answer was assigned the code *Improved answer*. A common situation for this code's appearance is:

- for classes, when a participant answered with a plural noun for describing groups (classes) in the first quiz and singular in the second quiz,
- for objects, when a participant explained that an object is a part of a group in any way (i.e. not a tangible thing), and
- for properties, when a participant both added the property hair colour (for instance) and expressed that properties is something that an object can do or is.

Following this coding procedure, the researchers thoroughly went through these codes and read the data once more. Finally, a conclusion was made by each of the researchers which was later compared and discussed between the two.

It is worth mentioning that the narrative voice in the artefact was an effective way of interacting with the users. This was observed throughout the experiments when none of the participants had any difficulties understanding the tasks in the game. It also provided the opportunity to implement a simple feedback mechanism to alert the player to follow the tasks or click inside the selected areas. Additionally, the voice mechanism was appraised by the school's teachers and organiser of CoderDojo, explaining it as being a creative way of connecting with the children.

Something that proved to be difficult in this study was to compare the quantitative and qualitative results with each other, in order to come to stronger conclusions. Since the only quantitative findings are the opinions of the participants (i.e. whether the game was fun or hard and if they learned anything), not much can be compared. But the researchers still triangulated the entertainment levels of the game, by comparing observation with the quantitative answers of the interviews, which clearly showed an indication that the game is entertaining for the majority of the participants. The conclusion would not have been as strong, if both types of collections would have existed for triangulation. The same applies for the results regarding the difficulty of the game and whether or not incidental learning was effectively implemented

B. Threats to Validity

The criteria used for evaluating validity are based on those proposed by Easterbrook [36].

Construct validity is a threat when the research design is vague and up for interpretation. After implementing the experiment and interpretation of data, it became clear that the artefact was not able to fully teach the concept of objects to the participants. The concept is rather abstract and hard to visually explain, especially to a child. The same goes for the pre- and post-quiz, they may not fully evaluate the participants knowledge about objects and how they are created. This potentially limited the findings during the experiment, where many participants showed signs of not understanding the concept.

Internal validity in this research can be assumed as when the participants become familiar with the outcomes of the questions in the experiment, or in other words, remember responses for the upcoming questions. This becomes particularly relevant when using a pre- and post-test. The solution in this study was to make sure to change the content of the questions, while keeping the same structure. Although, this does not remove the risk of the participants remembering the way they answered in the pre-test.

External validity in this study is mainly regarding the participants and their individual impact on the experiment. Since the involved participants in the study were young children, there is always a risk that they want to answer everything correctly and therefore compromising the results by giving answers they believe the researchers want, rather than answers that represents their perception. This can potentially affect the validity of the findings. Another threat is that not all of the

participants seem to have the same level of abilities, some may have lower IQ or lower level of skills of playing games. This can affect the overall results of the experiment.

Another external threat may be participants different levels of programming skills, since some of them had experience of playing other programming games and some did not. However, this risk did not seem to affect the answers of the tests, the concept of OOP is rather different than learning to program.

Finally, the quantitative and qualitative results may not be compatible for comparison. However, this risk was reduced by carefully deciding what variables to use for each of the two types of findings.

VI. CONCLUSION AND FUTURE WORK

The final results of this study indicates the positive feasibility of teaching the elementary concepts of OOP to children at the age of 7-12 years old. The findings also stated that the concepts of classes and properties are easier to grasp for children than the concept of objects. Thinking of objects in a way that is consistent with OOP on an abstract level, rather than thinking of them as tangible things seem to be rather challenging for most of the involved participants.

A rather overlooked method of acquiring knowledge is through incidental learning. This was achieved in most cases in the experiment, as the involved participants expressed that they did not know what was being taught. However, proving that this helped the children learn better is not possible with the current results. For this, a new study can be conducted that tries to explore the different approaches of teaching in regards to OOP that are mentioned in this report, this would lead to a more narrow field of investigation. Moreover, other notions of OOP such as inheritance, message passing and encapsulation may be added to the game. Due to the short span of this study, the artefact was not tested by many children. Hence a new study could be carried out to continue this research by experimenting with the artefact on a larger number of samples, with an equal number of participants for each age, for more solid results.

The final results indicates the artefact's level of entertainment. Only a few of the oldest participants found the game to be a bit boring or too easy. This was due to the large age gap of the participants. One of the goals of the study was to be able grab the players attention while playing the game and make sure they enjoy playing it. This aspect positively affected the learning outcomes of the game.

It was concluded that the designed artefact proved its capability and showed that after some minor upgrades, it is capable of explaining the notions of OOP to children at younger ages and provide the opportunity to let them think in an object orient manner. Considering the popularity of the object oriented style of programming in the industry, the designed artefact can be used as a beneficial tool to teach the basic concepts of OOP to children.

Some of the participants gave specific comments about how they think the game could be improved. Some of the children suggested that more freedom should be given to the user, thus

making the game more entertaining. Others stated that the game should be extended with more scenarios and increasingly harder tasks. These comments can be considered in future work to improve the quality, performance and usability of the game. Furthermore, not only can the game be improved in various ways, but a completely new tool can be made based on the results of this report. This derives from the fact that the findings indicated that children can learn OOP concepts, and since most of the children had learnt other types of programming such as Scratch and Hour of Code, they are capable of both actual programming and OOP design. Therefore, we believe that it is possible to teach OOP in a practical manner as well. This can be done, for instance, by combining the approach of the artefact of this study with the approach taken by the creators of Scratch.

ACKNOWLEDGEMENT

In this report we want to thank Dave Stikkolorum and Michel Chaudron, our supervisors, for their help in this study. Their guidance is crucial for the success of the research.

Additionally, we want to thank Kronaskolan, CoderDojo and the children that participated in the experiments. Without them, this study would have been lacking, and the results would not be as strong.

Lastly, Ida Årstein, a bachelors student of *art of pre-school teaching* at the university of Gothenburg aided this research by commenting on the design of the artefact, and contributed by narrating the story of the game.

REFERENCES

- [1] C. Xiajian, W. Danli, and W. Hongan, "Design and implementation of a graphical programming tool for children," *IEEE International Conference on Computer Science and Automation Engineering*, vol. 4, pp. 572 – 576, 2011.
- [2] Y. B. Kafai, "Software by Kids for Kids," *Communications of the ACM*, vol. 4, pp. 38–39, April 1996.
- [3] May 2017. [Online]. Available: <https://scratch.mit.edu>
- [4] May 2017. [Online]. Available: <http://www.alice.org>
- [5] May 2017. [Online]. Available: <https://www.greenfoot.org>
- [6] C. Britton and J. Doak, *A Student Guide to Object-Oriented Development*. Elsevier Ltd., 2005.
- [7] C. T. Wu, *A Comprehensive Introduction to Object-Oriented Programming with Java*, 1st ed. New York, NY, USA: McGraw-Hill, Inc., 2008.
- [8] B. Meyer, *Object-Oriented Software Construction*, 2nd ed. Interactive Software Engineering Inc. (ISE) 270 Storke Road, Suite 7, Santa Barbara, CA 93117, USA: ISE Inc., 1988.
- [9] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Lopez, J. Loningtier, and J. Irwin, "Aspect-oriented programming," *ECOOP'97 — Object-Oriented Programming programming*, vol. 1241, pp. 220–242, 1997.
- [10] I. Krajnović, L. Bakić-Tomić, and V. Markovac, "Object oriented programming in primary education," The Faculty of Teacher Education, University of Zagreb, Ulica kralja Zvonimira 8, 10000, Zagreb, Croatia, Tech. Rep., 2007.
- [11] T. Rentsch, "Object oriented programming," *SIGPLAN Not.*, vol. 17, no. 9, pp. 51–57, Sep. 1982.
- [12] A. C. Kay, "History of programming languages—ii," T. J. Bergin, Jr. and R. G. Gibson, Jr., Eds. New York, NY, USA: ACM, 1996, ch. The Early History of Smalltalk, pp. 511–598. [Online]. Available: <http://doi.acm.org/10.1145/234286.1057828>
- [13] I. E. Sutherland, "Sketchpad: Man-machine graphical communication system," 1963.
- [14] G. M., *Open Verification Methodology Cookbook*, 1st ed. Springer Science+Business Media, LLC, 233 Spring Street, New York, NY 10013, USA): Springer, 2008.
- [15] S. .D and M. Fayad, "Object-oriented application frameworks," *ACM*, vol. 40, no. 10, pp. 32–38, October-1997.
- [16] J. Paul Gee, *What video games have to teach us about learning and literacy*. Palgrave Macmillan, 2002.
- [17] A. B. Begel, "Bongo: A Kids' Programming Environment for Creating Video Games on the Web," *Games on the Web. Electrical Engineering and Computer Science Department*, vol. 1, pp. 1–87, May-1997.
- [18] S. Papert, *Mindstorms: children, computers, and powerful ideas*, 14th ed. New York, NY, USA: Basic Book, Inc., 1980.
- [19] D. H. Clements and D. F. Gullo, "Effects of computer programming on young children's cognition," *Journal of Educational Psychology*, vol. 76, p. 1051–1058, 1984.
- [20] B. S. Bloom, *Taxonomy of educational objectives: The classification of educational goals*. Longmans, Green & Co, 1956.
- [21] L. Anderson, D. Krathwohl, P. Airasian, K. Cruikshank, E. R. Mayer, P. Pintrich, J. Raths, and M. C. Wittrock, "A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives Complete ," *The Quest for Strengths: The Dawn of a Talent-Based Approach to K-12 Education*, vol. 83, no. 3, pp. 154–159, 2005.
- [22] J. L. Sherry and A. Pacheco, "Matching Computer Game Genres to Educational Outcomes," *Electronic Journal of Communication*, no. 1, pp. 214–226, 2010.
- [23] M. Kölling, "The greenfoot programming environment," *Trans. Comput. Educ.*, vol. 10, no. 4, pp. 14:1–14:21, Nov. 2010.
- [24] W. Dann, D. Cosgrove, D. Slater, D. Culyba, and S. Cooper, "Mediated transfer: Alice 3 to java," in *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '12, 2012, pp. 141–146.
- [25] S. Cooper and R. Dann, W. Pausch, "Teaching Objects-first In Introductory Computer Science," *ACM*, vol. 35, no. 1, pp. 191–195, January-2003.
- [26] May 2017. [Online]. Available: <http://snap.berkeley.edu>
- [27] L. Laporte and B. Zaman, "Informing Content-driven Design of Computer Programming Games: a Problems Analysis and a Game Review," *ACM*, no. 61, pp. 1–10, Oct-2016.
- [28] B. L. Mitchell, *Game Design Essentials*. Wiley, 2012.
- [29] C. Watson and F. Li, "Game-based concept visualization for learning programming," *ACM - MTDL '11 Proceedings of the third international ACM workshop on Multimedia technologies for distance learning*, no. 433117, pp. 37–42, Dec-2011.
- [30] J. M. Wing, "Computational thinking," *Communications of the ACM*, vol. 49, pp. 33–35, 2006.
- [31] A. R. Hevner, S. T. March, and S. Ram, "Design Science in Information Systems Research," *MIS Quarterly*, vol. 28, no. 1, pp. 75–105, 2004.
- [32] J. W. Creswell, *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches*, 4th ed. Mathura Road, New Delhi 110 044, India: Sage Publications, Inc., 2014.
- [33] C. Shaw, L.-M. Brady, and C. Davey, "Guidelines for research with children and young people."
- [34] B. D. Wright and M. H. Stone, "Best Test Design.Rasch Measurement." pp. 10–140, 1979.
- [35] D. R. Stikkolorum, M. Chaudron, and O. Bruin, "The Art of Software Design, a Video Game for Learning Software Design Principles," *Institute of Advanced Computer Science, Leiden University Niels Bohrweg 1, Leiden, the Netherlands*, pp. 1–4, 2014.
- [36] S. Easterbrook, J. Singer, M. Storey, and D. Damian, "Selecting empirical methods for software engineering research," in *Guide to advanced empirical software engineering*, p. 285– 311, 2008.

APPENDIX

A. The game in a sequence

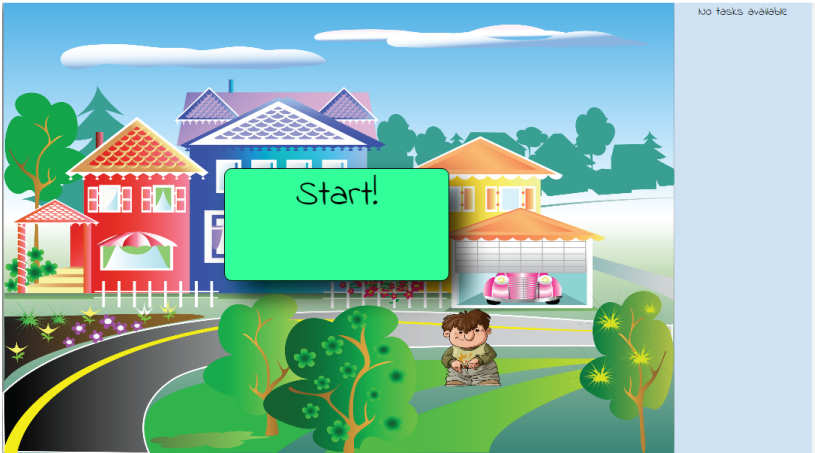


Fig. 15. Main menu of the artefact

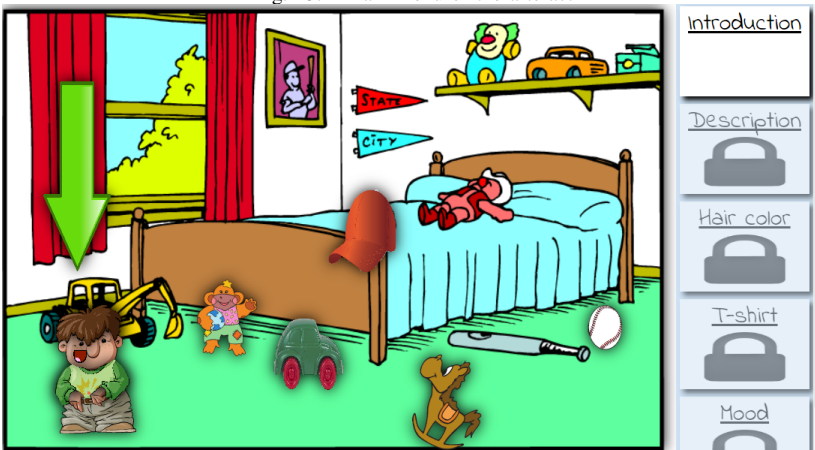


Fig. 16. Scenario 1 - first task: click on Kim

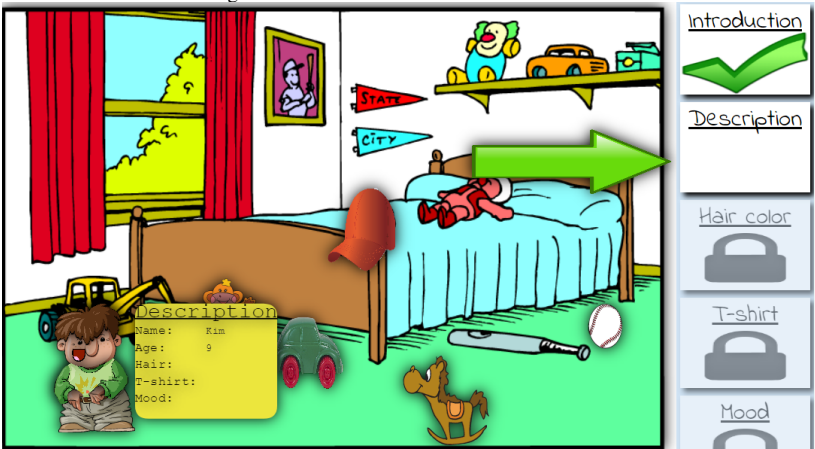


Fig. 17. Scenario 1 - second task: click on any object

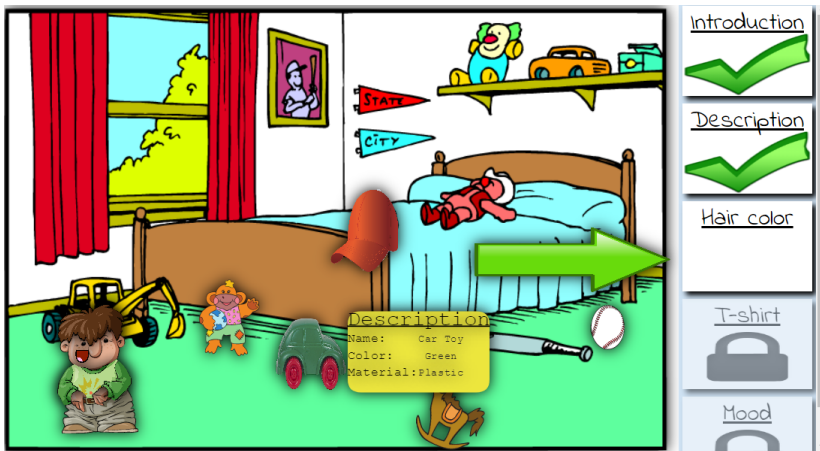


Fig. 18. Scenario 1 - second task: finished



Fig. 19. Scenario 1 - third task: change hair color

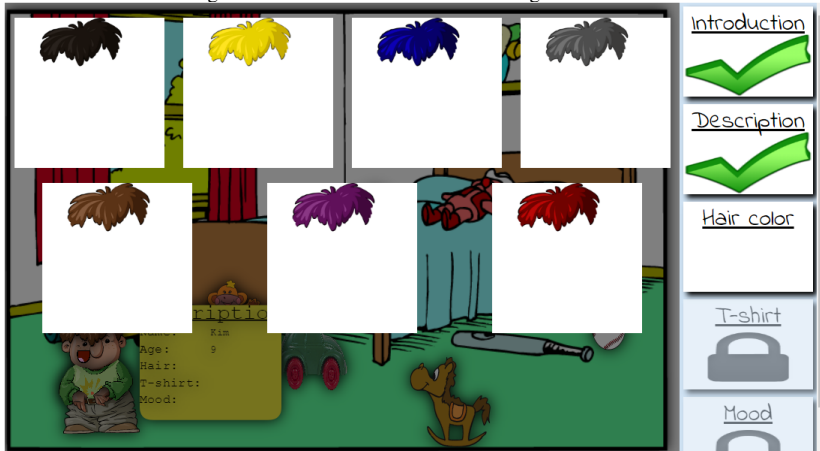


Fig. 20. Scenario 1 - third task: property chooser window

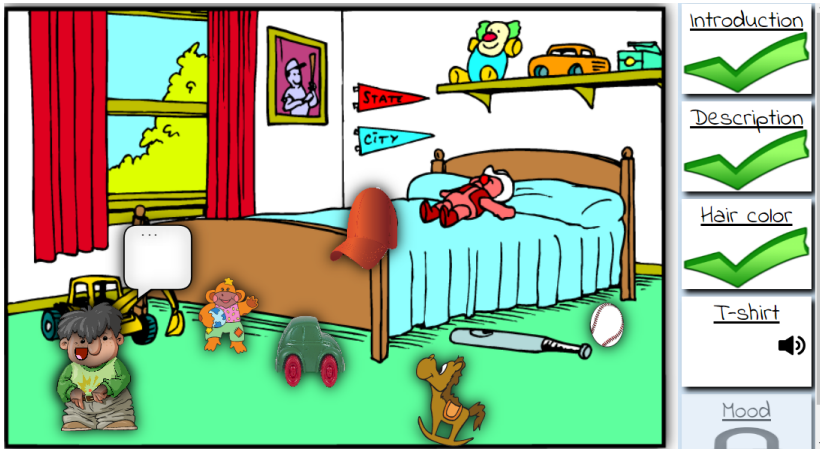


Fig. 21. Scenario 1 - fourth task: playing



Fig. 22. Scenario 1 - fourth task: choosing t-shirt



Fig. 23. Scenario 1 - fifth task: choosing mood (mouth)

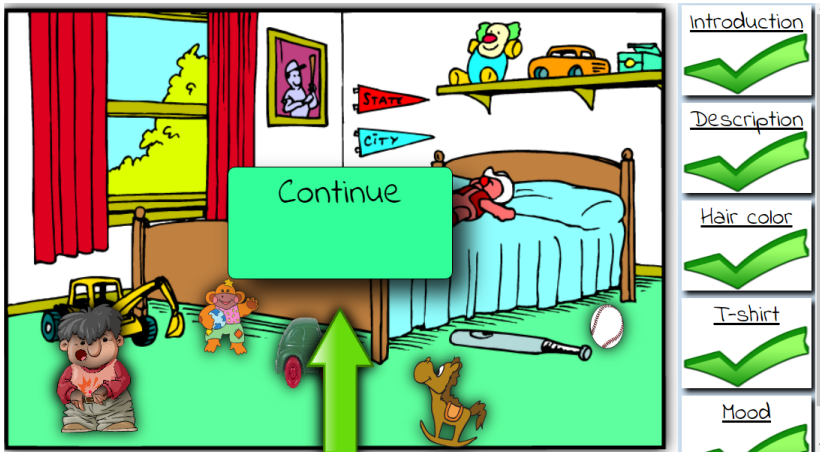


Fig. 24. Scenario 1 - scenario done



Fig. 25. Scenario 2 - first task: inspecting the family's properties



Fig. 26. Scenario 2 - second task: playing



Fig. 27. Scenario 2 - second task: adding a property



Fig. 28. Scenario 2 - second task: property added



Fig. 29. Scenario 2 - scenario done

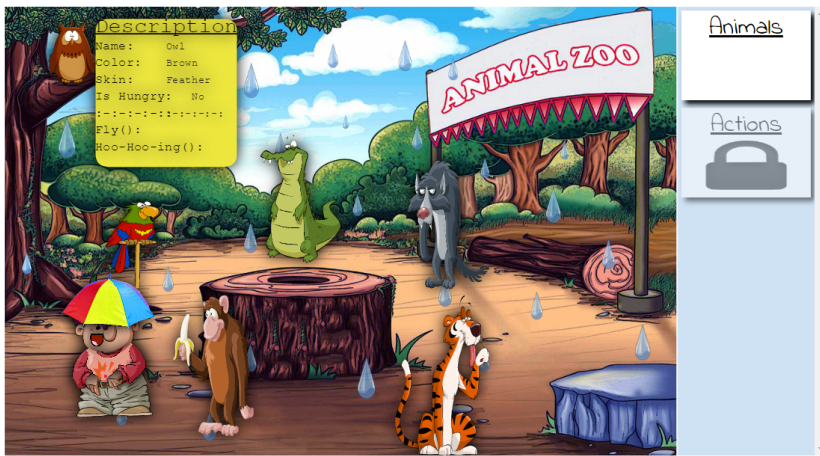


Fig. 30. Scenario 3 - first task: inspecting the animals' properties




Fig. 31. Scenario 3 - second task: playing



Fig. 32. Scenario 3 - scenario done

B. The Design of the Pre-quiz

1. Can you try to match the photos that belong together? Draw a line between all images in the same group. There may be several groups. Can you also write the name of the groups in the green box?



1.


2.

3.

Fig. 33. Pre-quiz, question 1

2. Can you fill in the blank sections (the dotted lines) in the green box below by looking at the image? Follow the example:

a) Example:



Description


Color: Yellow

Material: leather and fabric

Sport: Tennis

Shape: circle

b) Test yourself:



Description

name:

T-shirt


Color:

Age:

Fig. 34. Pre-quiz, question 2

3. Can you write a name for the group of pictures below?

Group Name:



4. Which of the pictures below can also be placed in the same group as in the above (last question)? Circle your answer.


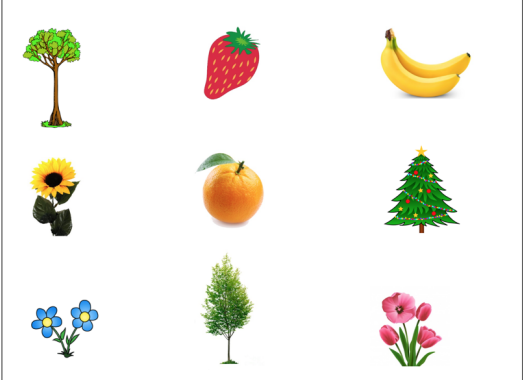


Fig. 35. Pre-quiz, question 3

C. The Design of the Post-quiz

1. Can you try to match the photos that belong together? Draw a line between all the images in the same group. There may be several groups. Can you also write the name of the groups in the green box?



1.


2.

3.

Fig. 36. Post-quiz, question 1

2. Can you fill in the blank section (dotted lines) in the green box below by looking at each image?

a)




Description

Name:

Taste:

Colour:

b)



Description

Sport:

Material:


Color:

Shape:

Fig. 37. Post-quiz, question 2

For the next task, can you fill in the green description box?
Can you also try to add a new line to the description of the image? write it on the empty line.

d)



Description

Name:

Age:


Mood:

..... :

Fig. 38. Post-quiz, question 3

3. Can you write a name for the group of images below?
Also fill in the green description box.

Group name:



Answer:

Group Description


1. Speed

2.


3.

Fig. 39. Post-quiz, question 4

4. Which one/s the images below belong to the group in the last question? Circle your answer.



5. Can you think of three things that are different in the two images below? You can write your answer in the green description box.



1.

2.

3.

Fig. 40. Post-quiz, question 5

6. Description of a group "CAR" is shown in the blue box on the left. Can you, through the information from the group CAR, make a new object (CAR 1). You can write your answer in the green description box.

Description for the group CAR

Name

Colour

Speed

.....

Drive()

Break()

➔

CAR 1

Name:

Colour:

Speed:

.....

Drive()

Break()

Fig. 41. Post-quiz, question 6