Learning Object Oriented Programming Using Augmented Reality

A Case Study with Elementary School Students

by

Tanvi Patel

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved May 2017 by the
Graduate Supervisory Committee:

I-han Hsiao, Chair
Brian Nelson
Erin Walker

ARIZONA STATE UNIVERSITY

August 2017

ABSTRACT

There is a demanding need to empower students from kindergarten through high school to learn computer science and be equipped with the computational thinking skills that they need in today's technology driven world. However, introducing computer programming to students can be challenging, especially for those who aren't familiar with the nuances of code. Several popular tools are used in curriculum for K-12 students which utilize interactive and visualization approaches to engage young kids in learning computational concepts. Possibilities of using Augmented Reality (AR) in teaching programming to novices are explored in this work.

In this thesis Ogmented, an AR application is designed which includes interactive learning material that covers a range of fundamental Object-Oriented Programming (OOP) concepts. This work aims to exploit the idea to learn abstract concepts via AR by capitalizing the strength of visual-aided and interactive elements. A user study with a group of elementary school students is conducted. It explored how students operated the AR application with the interactive elements and how they wrote codes to solve programming problems. It was observed that students who followed instructions while taking tutorials were successfully able to write fragments of codes in exercise modules. Irrespective of their knowledge about programming, majority of students were able to write executable code snippets for concepts they were taught with use of Ogmented. This shares an initial insight on using AR in classroom to teach abstract programming concepts.

DEDICATION

To loving family and friends

ACKNOWLEDGMENTS

I would like to take this opportunity to thank my thesis advisor Dr. Sharon Hsiao. She has been very approachable and supportive. I am extremely grateful for her prompt replies, encouragement and her feedback of all kind. Being mentored by her was a wonderful learning experience. I would like to thank to Dr. Brian Nelson and Dr. Erin Walker for lending their invaluable time to be a part of my committee. I appreciate Prof. Elva S.Y. Lin's efforts for connecting me to subjects of this user study. She is been doing great work by giving hands on experience of technology to elementary and middle school students.

I want to thank Dr. Kurt VanLehn and Dr. Jon Wetzel for giving me an opportunity to be part of FACT team. It has been a privilege to work here. I would also like to thank my family and extended family for their support in all forms. Thanks to my friends for making this journey fun and interesting. I would like to give special credits to Dikshay Poojary with whom I developed first AR application in last summer. His constant motivation and support is much appreciated.

TABLE OF CONTENTS

Page

CHAPTER Page

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

"Computer Science for all" is an initiative by former president of United States that was taken to empower a generation of American students with the computer science skills they need to thrive in a digital economy. This initiative focused on empowering all American students from kindergarten to high school to learn computer science and be equipped with the computational thinking skills. Because it is essential for them to not just be consumers but to be creators in the digital economy and to be active citizens in technology-driven world.

90 percent of parents in the U.S. want their children to learn computer science as it impacts every career in today's world. But only 40 percent of 7th-12th grade schools teach it [21]. A focus on STEM is not enough: while 70 percent of new STEM jobs are in computing, only 7 percent of STEM graduates are in computer science [16, 17]. It is essential that savvy schools begin to focus some STEM resources on computer science and programming.

1.1 Motivation

To successfully implement involvement of computer science in curriculum of K-12, associated challenges need be overcome. Teaching computer science concepts to students do not come easy. K-12 teachers need resources, starting with a compelling definition and examples to explain concepts in way that students at their age understand.

There are some tools and software that help in teaching younger kids how to code. Tools like Scratch [26] and Alice [27] follow game based and storyboarding approach of teaching. Visual aided interactive approaches are used for better effectiveness and faster learning. Research reported in educational literature suggests that using visuals in teaching results in a greater degree of learning [3]. In order for visual enhancements to be used most effectively, teachers are expected to possess skills that include the language of imagery as well as techniques of teaching visually.

Learning programming concepts and computational thinking becomes a tedious process considering abstract and complex nature of it. Pertinent and adapted feedback should be made available for various components of programming tasks. To help elementary students and novices for better learning experience presentation of these concepts has to be made available such that they can relate to while learning. In this work, the idea of using interactive 3D UI coming live in physical world to learn OOP in elementary school is explored.

1.2 Research Questions

This thesis explores the possibilities of using Augmented Reality (AR) in teaching programming to novices. AR works by rendering virtual objects in real physical world. Unlike immersive Virtual Reality, AR interfaces allow users to see the real world at the same time as virtual imagery attached to real locations and objects. And therefore AR enhances real world experience unlike other computer devices that gets users immerse in screen by taking them away from real world.

Though there are some popular tools around to teach programming concepts to K-12 grades, use of AR could open new doors to learning by getting learning to live physical world. There is no work in existence as of today that use AR technology to teach object-based educational programming. There are several AR based learning tools available for students to learn topics from subjects like chemistry, biology, mathematics. Ogmented - an AR based application to perform a study on elementary schoolers to learn OOP is created in this thesis. Ogmented teaches some of abstract programming concepts like object creation, method invocation, method binding etc. This thesis analyses learning pattern of users of Ogmented.

This work addresses following research questions:

1) Is Ogmented useful and engaging to teach programming to novices?

2) Is there a learning pattern in user interactions with Ogmented while they take tutorials to learn abstract programming concepts?

3) Is there a pattern of writing code by participants? Is Ogmented helpful in teaching how to code?

4) What are some common errors made by participants of user study in exercise modules? Is there an association between pattern of most common errors and user learning pattern?

5) Is there any relation between learning in tutorials and performance in exercises when using Ogmented?

CHAPTER 2

BACKGROUND

2.1 Students in Computer Science

Seventy percent of parents surveyed in a research study [7] said they would also like to see their local schools spend more money on up-to-date and well-equipped science labs, more equipment for hands-on learning (69%), and more equipment to help students learn computer and technology skills (68%). The majority of parents with children in Grades 6–12 said that they want to see more emphasis in their child's school on STEM topics, such as computer programming (65%), basic engineering principles (52%), and statistics and probability (49%). More and more parents in United States want their children to learn programming and logic, though factual data doesn't reflect the same.

A 2012 report by the President's Council of Advisors on Science and Technology (PCAST) predicts that the U.S. workforce will suffer a deficit of one million college graduates in science, technology, engineering, and mathematics (STEM) over the next decade [6]. There is not enough of supply considering the dependency on computer technology and availability of computer science and information technology related jobs. Another interesting thing to note is that out of all students those enroll themselves in STEM education, do not make it through. Less than half of the three million students who enter U.S. colleges yearly intending to major in a STEM field persist in STEM until graduation [6]. Many educationalists focus on increasing the persistence rate. Some of the

factors that are involved in motivating students to drop out are directly related to their engagement and understanding of concepts taught in class.

To learn programming is one of the foundations of computer science. Students are expected to come up with logical solutions that they code to solve problems. In schools, curriculums are designed in a way that they focus on starting with introductory programming course in CS program. However, regardless of the recognized importance of learning programming, the results are often disappointing. Several multi-institutional studies [8, 9, and 10] have indicated that there are serious deficiencies in the learning outcomes of students who have passed one or several programming courses in CS programs. These problems originated from misconceptions on early studies. Poor understanding of basic concepts, procedures and processes is a poor basis for advanced studies.

Programming concepts can be overwhelming for students to learn and teachers to teach. Sometimes it can lead to lack of interest for students in learning how to code as despite of spending more time they do not understand introductory concepts clearly. Statistics show that the number of Computer Science majors is dropping across the United States [4]. Research shows that there are multiple reasons associated with it. One of which is that many students find it difficult to learn computer programming. In many cases failure in the programming module is sufficient for a student to be required to withdraw from the course and leave the University. This is clearly not a satisfactory situation for either the students or the School [5]. A statistical analysis (Iain et al. 2002) shows that topics that rely on a clear understanding of pointers and memory-related concepts (such as copy constructors and virtual functions) prove to be the most difficult for students to

understand. Analysis claims that these concepts are only hard because of the student's inability to comprehend what is happening to their program in memory, as they are incapable of creating a clear mental model of its execution.

2.2 Challenges in Learning Programming

Learning programming can be challenging because it includes not only learning syntax of language but also involves logic clarity and conceptual understanding. Talking about some of the challenges involved in learning programming, first and foremost, students need to have clear understanding of constructs available in programming language. Many novices and students who are learning to code have fuzzy notions about language basics. Misunderstanding or not enough understanding of construct of language lead to confusion and prevent learners from performing well.

Other area of research is to study student's participation in learning. An active participation rather than passive participation has been proven to be more effective while teachers teach fundamentals and semantics of programming. There is evidence that a participative approach may be effective in higher education. For example, in a computing subject area, Fleury [11] shows how complicated recursive algorithms can be presented effectively to students using an approach whereby the students themselves enact the processes involved. Aside from teaching details of the syntax and semantics of a particular programming language, Soloway argues [12], it is necessary to explicitly and concurrently explain why and how programs work, the goal of any given program, what plan segments are, strategies for decomposing tasks, rules that well-formed programs

adhere to, and design strategies. Abstract nature of some concepts adds difficulties in understanding to learners.

Overall challenges with learning traditional programming language include following:

- To write code is difficult task. Early programming languages were difficult to use. Students could not master syntax of programming [22].
- Teaching programming was not compared to young people's interests or experience.
- Development and enhancement of computational thinking to write code is integral part of learning as it focuses on process of abstraction.[25]

Numerous studies argue that students view computer programming as a purely technical activity rather than a set of combined problem solving skills [23]. Therefore, the majority of students who are learning introductory computer programming tend to develop superficial knowledge and fail to create problem solving strategies through using programming constructs. Wing (2006) defines computational thinking as a set of intellectual and reasoning skills that states how people interact and learn to think through the language of computation. In other words, thinking computationally involves using methods, language and systems of computer science in order to solve problems in any discipline regardless of where the problem lies.

Programming and computational modelling are core embodiment of learning CS. Computational thinking can be integrated with K-12 science classrooms via visual programming [24].

2.3 Tools for learning programming for elementary schoolers

In recent years, new attempts have sought to introduce programming to children and teens [19]. Some use professional programming languages while others use new languages such as Alice developed specifically for younger programmers. Another popular work is Scratch [26] which is a visual programming language. With Scratch, programmers can create animated stories and informational texts. It is also used to learn topics in math, history, and even photography. Scratch flexibility allows teachers to create conceptual and visual lessons and science lab assignments. Despite this, Scratch does not provide every construct available in languages like Java. Nor does it support data types, data structures, methods, parameters, return values, inheritance, or polymorphism, all of which might be appropriate to introduce in introductory courses [20].

As mentioned earlier, Alice [27] is created for purpose of learning for younger programmers. It teaches object-based educational programming language with an integrated development environment (IDE). It uses a drag and drop environment to create computer animations using 3D models. It teaches fundamentals of programming concepts in context of creating animated movies and video games. In Alice, 3D objects are created which fill virtual world. Students write program to animate objects. In a controlled study it was observed that students with no prior programming experience taking their first computer science course rose average grade from C to B, and retention rose from 47% to 88% [21].

Upon realizing that OOP should be taught to elementary and middle school children in a way that would reduce complexities that come with learning programming and coding, some educationalists and researchers are constantly working on creating ways to make learning of programming to K-12 students effective and simpler.

2.4 Augmented Reality for Learning

As per Wikipedia [9], augmented reality (AR) is a live direct or indirect view of a physical, real-world environment whose elements are augmented (or supplemented) by computer-generated sensory input such as sound, video, graphics or GPS data. Ironman movie and Pokémon Go games are well known for illustrating AR. It is a growing field of technology where real life is modified and enhanced by computer-generated sights and sounds. The most common use of AR can be seen through mobile apps. By pointing device's camera at something that the app recognizes, and it generates a 3D animation or video superimposed over whatever is on camera's screen. The effect makes the computer-generated item appear like it's really there.

AR has its root developed in field of education and learning. Applications like Augment [28] and ELEMETS4D [29] provide a platform to start creating 3D content to explain concepts. Augment claims that by creating 3D models in physical world, it makes a complete learning cycle. Students retain more knowledge for a longer period. AR has been implemented for improved learning in field of Chemistry, Physics and Biology. I intend to study effectiveness of AR in teaching programming concepts to students. As AR can likely be an effective visualization tool to teach in class.

Following are some of the well-known AR applications used in classroom or outside of classroom for learning:

ELEMETS4D [29]: Elements4D provides an immersive and interactive way to learn Chemistry in classroom. Elements 4D is a set of interactive blocks that help students learn the Periodic Table by showing how elements combine into new chemical substances, what the reactions look like, and the resulting chemical equation. For chemical reactions that are too dangerous to complete in lab can be experimented with AR using this app. One of the testimonials says that students were really engaged in learning Chemistry this way and they wanted to learn more. This app got learning to life and opened up a whole new level of learning to Chemistry students.

ALIVE STUDIOS [14]: Alive Studios help early learners become proficient in reading and math by 3rd grade by equipping teachers with engaging AR technology. A case study [14] with this tool showed that students who typically had attention difficulties were consistently engrossed in the program. There was 48% increase in letter naming fluency and 112% increase in letter sound fluency. Alive studios are successfully winning reading and Math proficiency battle.

AURASMA [30]: It is a popular tool for creating and exploring AR experiences. The app works with triggers that teachers and students create on the web with Aurasma Studio.

Users can upload trigger images of their choice and add videos to make their very own augmented reality experience.

AR applications like these are changing future of classroom technologies. The first AR prototypes were created by computer graphics pioneer Ivan Sutherland and his students at Harvard University and the University of Utah, appeared in the 1960s and used a see-through HMD2 to present 3D graphics (Sutherland, 1968). The technological demands for AR are much higher than for virtual environments or VR, which is why the field of AR took longer to mature than that of VR. However, the key components needed to build an AR system have remained the same since Ivan Sutherland's pioneering work of the 1960s.

CHAPTER 3

OGMENTED

Ogmented is an AR based Android application. It is developed in this thesis to explore possibility to learn abstract programming concepts for novices. It explores learning effect of possibility of having 3D models in real physical life. In this thesis, Ogmented specifically focuses on *Object*. Ogmented is divided into two main sections- Tutorials and Exercises. Users are expected to first take tutorials to get equipped with the tool and coding concepts as well as syntax. Tutorials are designed to foster learning of programming concepts along with syntax by 3D rendering and visual programming. After finishing tutorials, participants can proceed to try out exercises. Aim of exercises is to test users on their learnings from tutorials. Exercises have different difficulty levels which will be used to analyze user learning patterns. Ogmented includes tutorial objects and exercises to teach object creation, method binding and method invocation.

3.1 Design Rationale

Purpose of having following design is to first enforce learning to users and later evaluate it. This helps in evaluating effectiveness of Ogmented as well as measure learning. Tutorials make users well equipped with how Ogmented works. Tutorials do not ask users to write any code, while exercises do. Having button executable actions could record false learning as users might engage in random action execution. Exercises therefore expect them to write code snippets to record their understanding of concepts and syntax. Effects of this approach of learning through Ogmented will be analyzed.

3.1.1 Tutorials

Tutorials are designed to let users get equipped with Ogmented and to teach them to create objects of specific class. It also lets them to interact with objects using methods. Objective of tutorials is to learn. While taking tutorials, expected behavior of users is to follow instructions presented to them and explore object creation. Object is created as an augmented element rendered on marker that can be seen through device screen. Predefined classes in Ogmented are Car, Butterfly and Zombie. Each class has up to 3 predefined methods. Upon creation of an object of specific class, methods available in that class are displayed to users in form of clickable buttons. Approximate time to take all tutorials is 5-6 minutes which differs from user to user.

Aim of designing following 3 tutorials is to progressively teach users to create objects and interact with them. It is essential that users interact with rendered objects to change their behavior and learn. Their ability to manipulate object behavior is logged for later analysis. Each tutorial is built with purpose to first teach object creation and interactions and later convert this learning to write code.

1) Tutorial 1 (Object Creation I):

In the first tutorial, object creation concept is embedded through interactive button click through augmented reality interfaces. Augmented objects will appear on click of the object-creation-button click. For instance, click on create a butterfly object; a butterfly

object will be displayed. Console panel (instruction panel) will show following code upon click:

Butterfly butterfly1 = new Butterfly();

Users can change behavior of created objects using methods. Methods can be invoked upon similar button click. Following syntax would be shown on console with button click of method *fly()* :

butterfly1.fly();

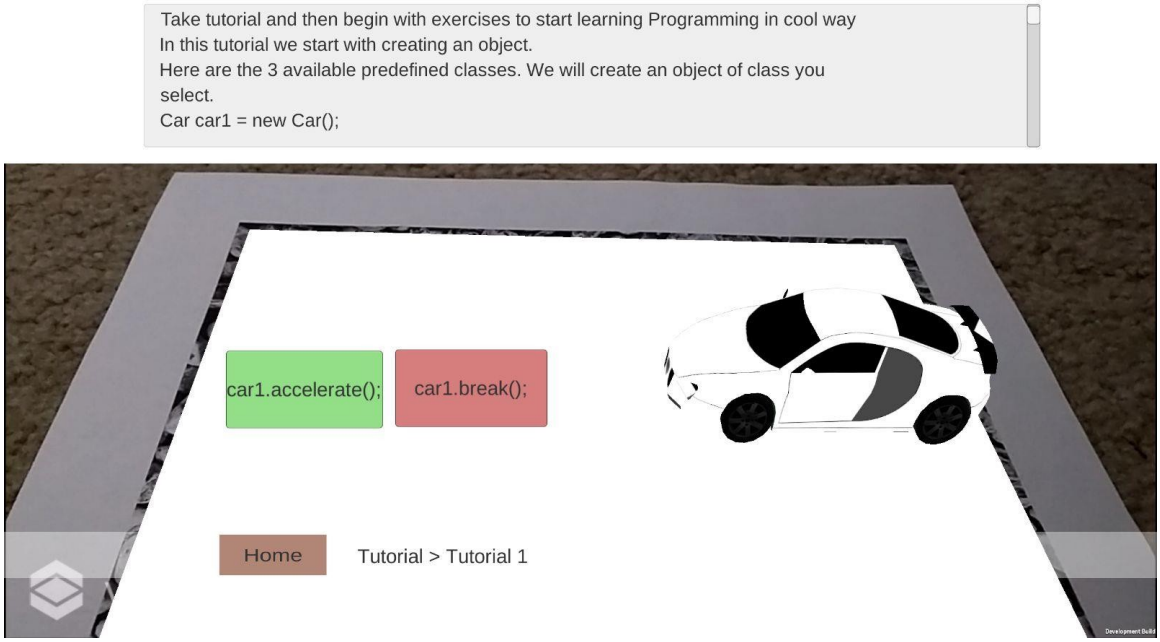


Figure 1. Tutorial 1 in Ogmented

Each pre-defined class is associated with two predefined methods (Table 1). The methods' availabilities are based upon objects' creation and will be presented to users accordingly. These methods are already declared in system of Ogmented. To invoke a method to change behavior of an object, following syntax is used:

objectTest.methodTest();

This syntax will invoke method *methodTest()* to change defined behavior of *objectTest*.

In tutorial 1 and 2, methods are presented to users in forms of buttons that can be clicked to interact with object. Users see code snippet related to action on console panel.

This is an introductory tutorial that teaches users to create object on click. Purpose of this tutorial is to take users through concepts and syntax of object creation and method invocation. Following console panel/ instruction panel messages would make them ready for creating single object and changing behavior of created object. Users are presented with the predefined classes. Object of selected class is created and displayed to users with available methods in vault for users in form of buttons. With 2 available methods available for each class, users can interact with object.

Class	Available Methods
Car	accelerate(), stop(), turn()
Butterfly	fly(), stop()
Zombie	walk(), stop()

Table 1. Available objects and methods in Ogmented

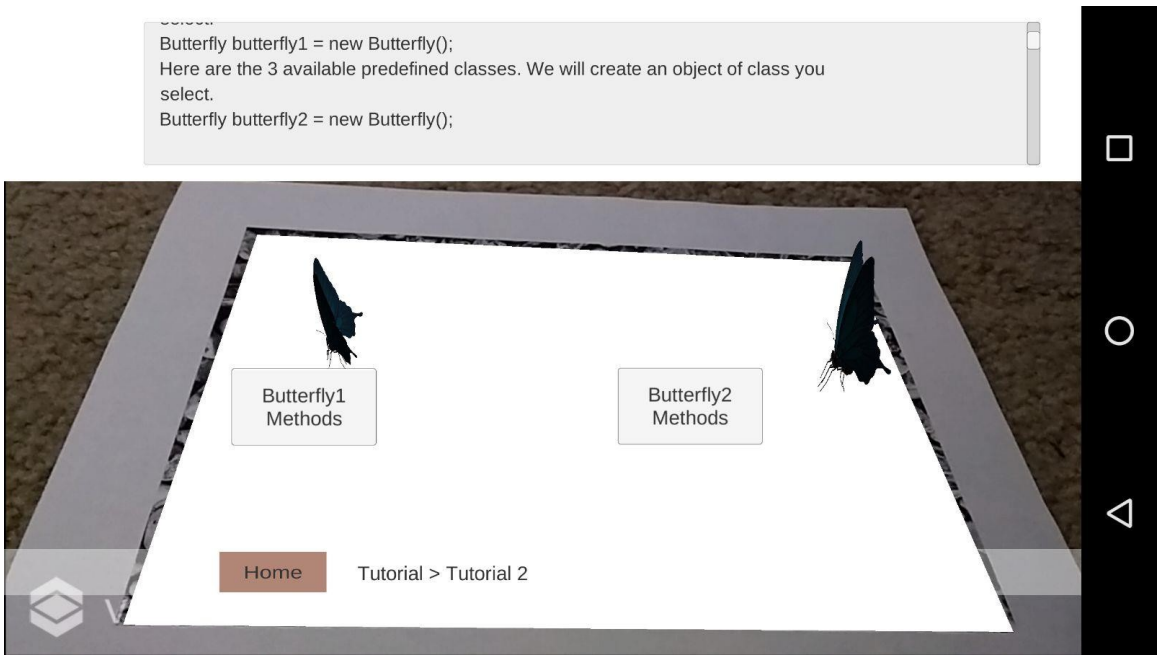


Figure 2. Tutorial 2 in Ogmented

2) Tutorial 2 (Object Creation II): In this tutorial, the core concept is to interact with multiple objects to change their behavior. Users can interact with multiple objects on screen. Just as in tutorial 1, objects can be created with object-create-button and methods can be called with invoke-method-button. Appropriate code is shown just like in previous tutorial.

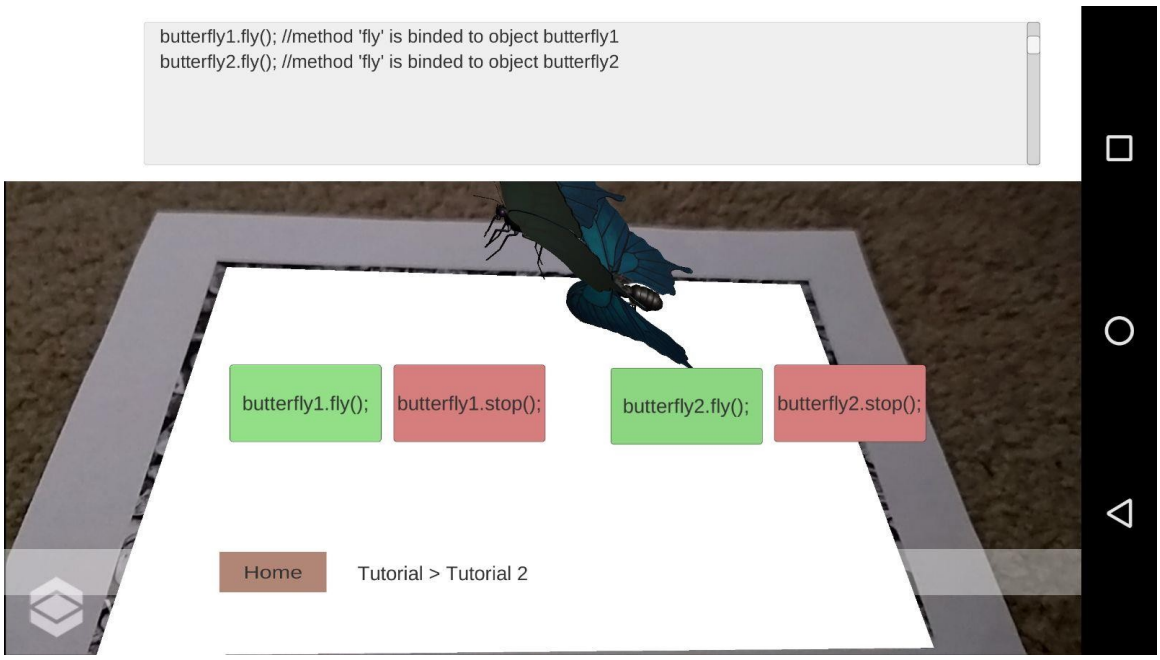


Figure 3. Tutorial 2 (with action buttons) in Ogmented

The aim of this tutorial is to make users understand that an object can be operated with methods bound to only that object. For example, object car1 can have car moving with car1.accelerate() command. In order to move object car2, accelerate() method has to be called on object car2. The assumption is that students' interaction sequences on object creation and method invocation will reveal the understanding of object behavior associations.

3) Tutorial 3 (Execute Code): This tutorial has a pre filled input field and a button to run code entered in this field. Butterfly object appears on screen with fly() method populated in input field. Users are expected to run the method in order to call prefilled method fly() on Butterfly object. Purpose of

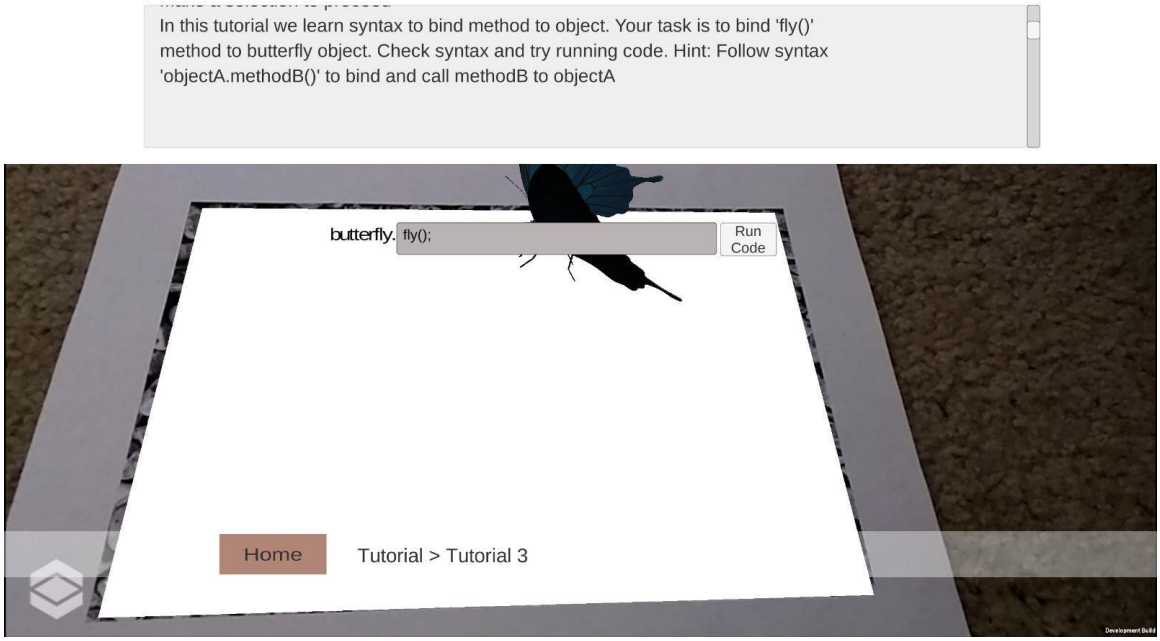


Figure 4. Tutorial 3 in Ogmented

Each tutorial comes with explanation of concepts that appear on top panel of application. Users are supposed to read instructions. By the time users are done taking all tutorials, they are expected to know following concepts:

- Object creation from predefined classes; syntax for object creation using “new” keyword
- Invocation of methods on objects; syntax for method invocation
- Knowledge of how method invoked on one object cannot do anything on other object present on screen.

3.1.2 Exercises:

Exercises have various sets of difficulty levels. Users are expected to start from exercise 1 and then proceed to next. As exercises start, instructions to execute exercises appear on instruction panel. After reading this, they are supposed to click “I am ready” button which then renders respective exercise scenes. Exercises in which code snippet is expected to be written and executed, as shown in Figure[x] hints about syntax of doing so which is displayed on information panel. Exercises contain simple tasks that users can perform on object by changing object attributes or by attaching appropriate methods. Approximate time to take all exercises is 8-12 minutes.

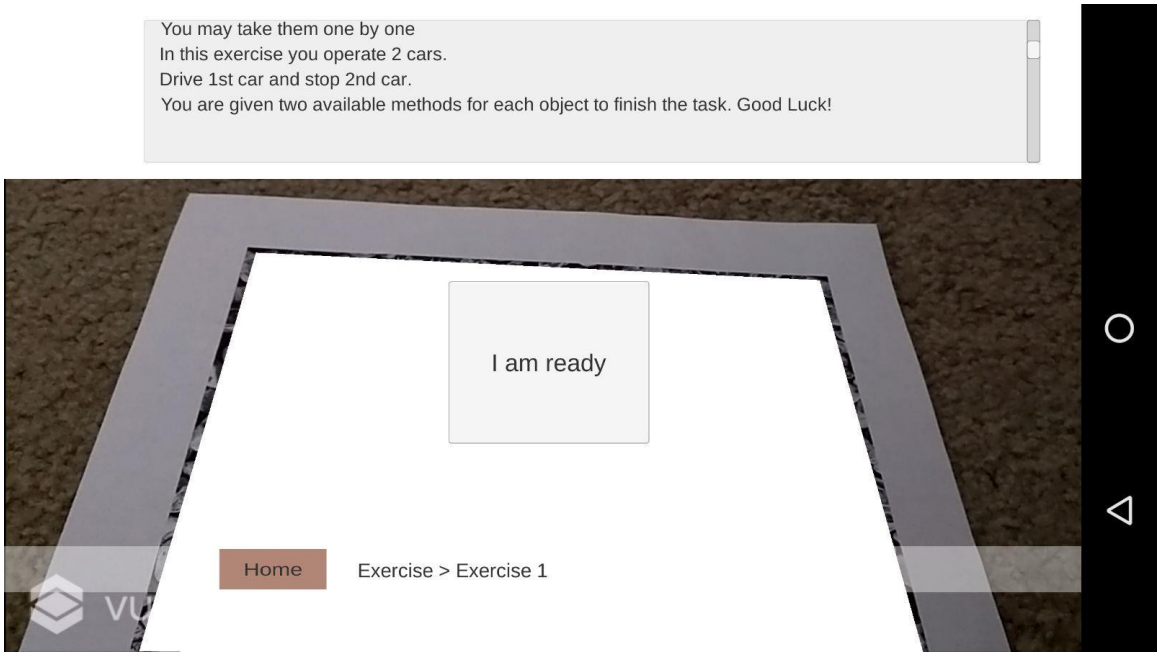


Figure 5. Exercise 1 in Ogmented

Exercises are to be reflected upon by users’ learning in tutorials. An expected behavior is presented to users. They are supposed to create objects and change their behavior by using declared methods in system to generate behavior expected in each exercise. Out of

3, 2 exercises require users to write code. Users tend to make syntax / semantic errors which is taken care. When this happens, they are notified with Syntax error/ Invalid Code message in console.

Analysis of exercise logs would reveal users' understanding of concepts taught earlier. Ogmented logs would be analyzed to evaluate effectiveness of the tool to test if it helps in learning of above concepts and code writing.

Exercise 1 (Drive Car I): This exercise focuses method invocation of same methods with two different objects. Exercise 1 doesn't involve writing of any code. This exercise is of low level difficulty. In this exercise, 2 objects of class Car are present which are car1 and car2. Users are supposed to drive car1 and stop car2. Methods associated with both objects are present on screen in form of buttons. Available methods with each objects are accelerate() and stop(). Users can call car1.accelerate() and car2.stop() in order to finish this exercise.

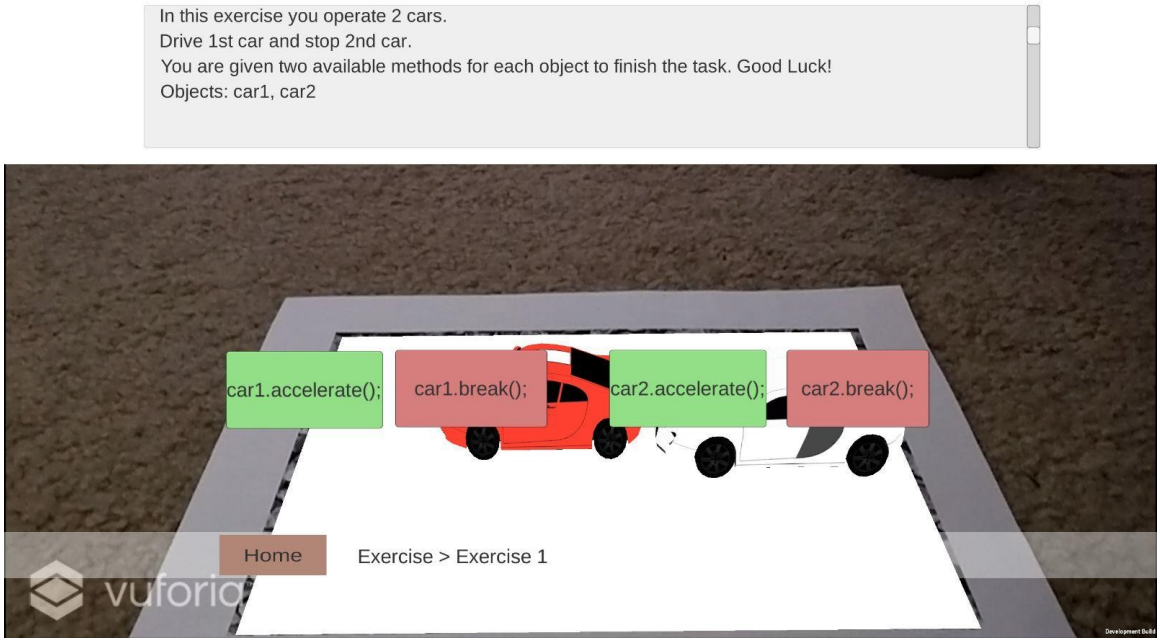


Figure 6. Exercise 1 (with action buttons) in Ogmented

This resonates with their learning of method binding from tutorial 1 and 2. This exercise should not take more than 1-2 minutes to finish. As soon as both exercise tasks are executed, a message in console notifies users that they are done with exercise and they could move to next one.

Exercise 2 (Drive Car II): This exercise tests users for their capacity of creating object. Users are supposed to write code and run it in order to successfully create object. Users are asked to create object of class Car using “new” keyword. After which users are asked to accelerate car and take turn using accelerate() and turn() method respectively. Instructions and preferred syntax of writing code on given exercise are put on information panel. Code is to be entered in code executer and ran by users to show result on rendering interface.

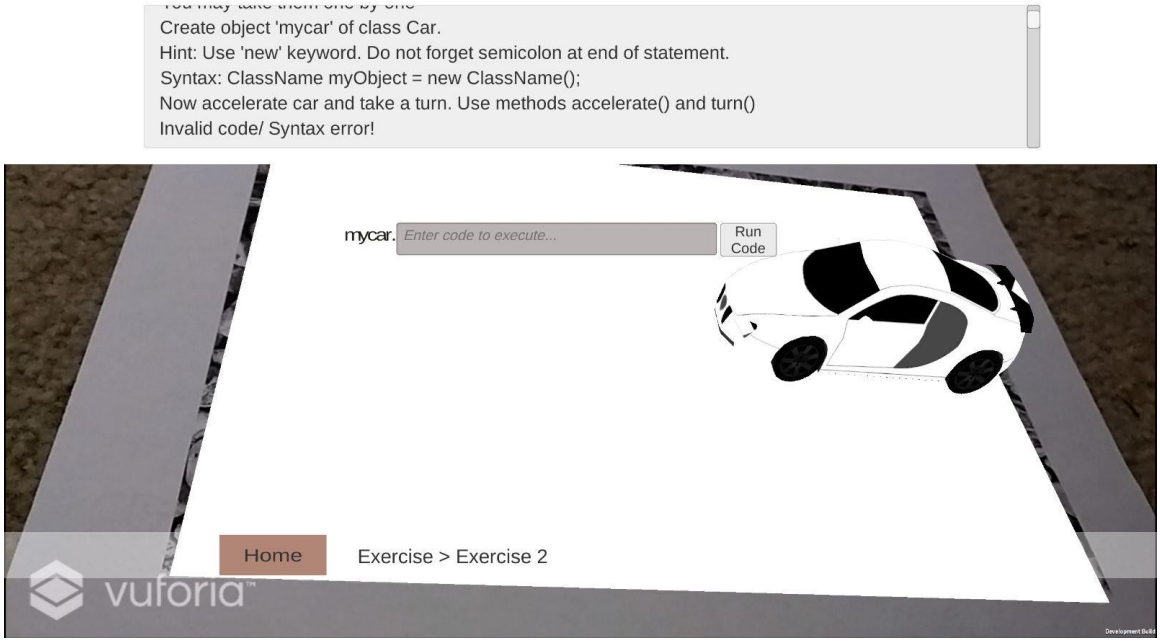
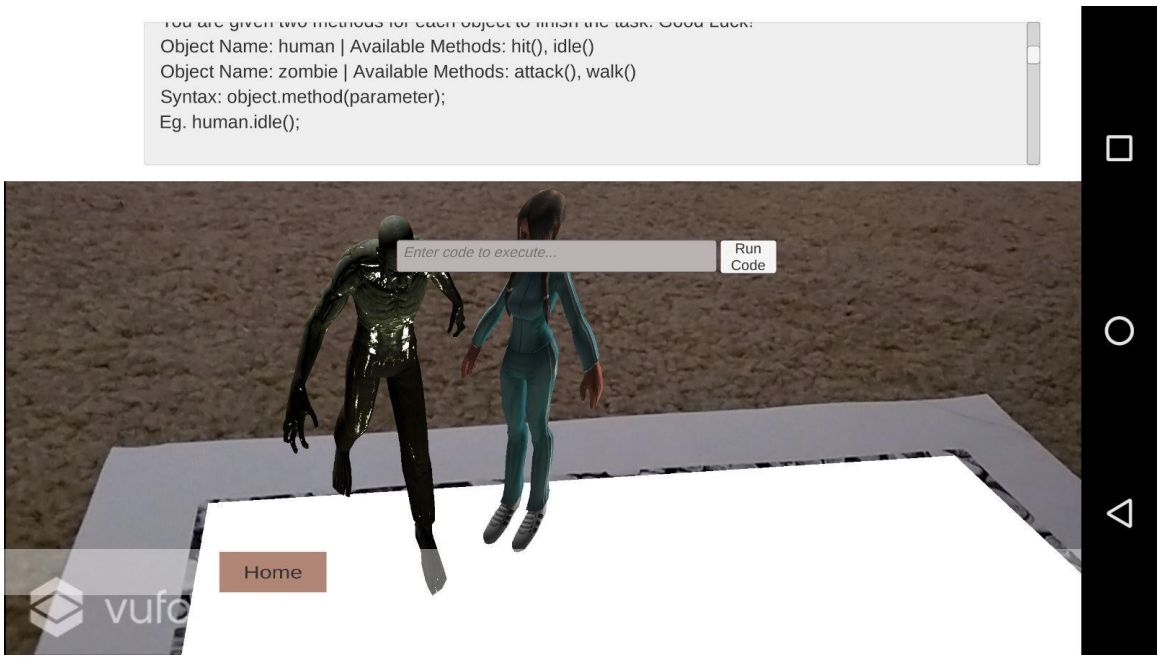


Figure 7. Exercise 2 in Ogmented

First 2 tutorials teach object creation with object-create-button click introduces and reinforces syntax to create new object using new keyword multiple times. In this exercise they are tested for writing and executing code snippets for the same. Their performance would be evaluated and analyzed for reflection of their learning from tutorial.

Exercise 3 (Kill Zombie): Exercise 3 consists of two objects; zombie and human with predefined methods kill(), walk() and hit(), idle() respectively for each object. Task in this exercise is to make human object kill zombie using available methods. Users are supposed to type in code and run to call methods on object. Users type human.hit() to successfully complete this exercise. Object human is introduced for first time in this exercise.



Exercise 8. Exercise 3 in Ogmented

3.2 Interfaces

Ogmented followed basic usability principles in user interface creation. There is a consistent interface present across all modules. By default, Ogmented opens in full screen mode in landscape orientation on devices. Interface is cross device compatible. Ogmented has following prime interface components:

3.2.1 Information panel/ Console panel

Ogmented interface consists of an information panel on top. Information panel adds new information as user progresses and navigates. It shows explanation on what user can expect from each exercise/ tutorial. This panel is used to educate users on how code is written for the action selected by him/ her. It is also called console panel as in case of any

wrong code written by user, notifications on syntax error or invalid code are added to panel text. Code and related comments with each button click shows in this the panel.

User is supposed to constantly look at panel for new information. Panel scrolls on its own upon addition of new information.

3.2.2 Rendering Screen

Rendering screen covers majority part of user interface. AR components are seen through rendering screen. Ogmented uses device camera to see elements present in surroundings. On scanning the marker it projects elements which are seen through the device.

3.2.3 Navigation breadcrumb

Breadcrumb is present in bottom of screen. Purpose of this component is to show how user navigated till present screen. This also has home button. Upon pressing home button user is navigated to home screen of app which is the selection page between tutorial and exercise.

3.2.4 Code executer

This component consists of a small input area where user can type in code statement. This comes along with “Run Code” button. Upon pressing this button, code is executed and ran. If code runs successfully, effected changes are made with app components. For example, on running car.turn(), car object in render screen is attached method turn which makes object car take a turn in physical environment. In case of any error, user is notified by error message being displayed on information panel.

CHAPTER 4

METHODOLOGY

4.1 Research Platform – Ogmented

Ogmented follows game based interactive learning approach to teach programming. Having visual augmented components make users see mistakes that they do while working with Ogmented. As majority of students and adults play games occasionally or frequently, this huge interest in game can be used to entice students to learn computer programming. Also, it was observed that over last summer a lot of students around became really interested in then popular Pokémon Go game. This was the motivation to develop Ogmented which took around 3 months' time. In this time Ogmented was designed keeping usability principles in mind. To not let users wander around to see the projections while studying, projection was confined to a small area. A marker was used that would act as QR code. This requires users/ participants to scan marker to proceed with AR object rendering. This marker is of size of A4 paper or a book. User interactions throughout each session were logged in application to later use this data for analysis.

4.2 Study Design

A user study was conducted with participants from elementary school to collect quantitative and .qualitative data. Participants were asked to take tutorials on Ogmented

platform and were tested against exercises that Ogmented has. Analysis was then performed on data collected to address research questions.

4.2.1 Audience

Study was conducted with 14 elementary students from 3rd and 4th grades who represented spectrum of usage behaviors. Participants spent nearly 15 minutes operating Ogmented.

4.2.2 Data Collection

User study was performed on 10 elementary students from 3rd grade out of which 3 were female students and 4 from 4th grades out of which 2 were female students and rest were male students. Participants were given Samsung Galaxy tab A/ Google Nexus 6 devices with Ogmented installed in devices. Participants spent average of 13.55 minutes to finish all tutorials and exercises. Data of their interactions were logged in application.

Logs saved following information:

Field	Description
User id	Unique user id generated by system
User click records	Includes button clicks or navigation records
Task id	Unique tutorial id / exercise id

Task related meta information	Other information related to tasks eg. Objects created, task name
User entered code	code entered by user for execution
Timestamp	Timestamp of each interaction

Table 2: Ogmented log details

Besides logs from Ogmented, participants were given a pre-survey form and a feedback form to fill. Pre-survey form focused on getting background information of participants. This included questions to know if participants were aware of OOP concepts. Purpose for this was to know if prior knowledge of programming concepts has any effects in learning. Feedback survey was designed to include any feedback that participants might have regarding the app and their experiences with Ogmented app as well as learning. Survey also included questions to ask if they would want to learn programming in future and if they were interested in learning how to code. This was included to check participant's inclination towards learning programming. As this could also play important part in learning process. Except for above methods, direct observations were noted.

4.3 Evaluation Metrics

To address the research questions, we define metrics to measure data qualitatively and quantitatively.

How users coded in exercises is essential to observe. Successful attempts in each exercise would explore if there exists a possibility of having a correlation between learning and performance. It is also essential to know if participants found Ogmented useful. This

would be determined by their ability to learn coding. In addition to qualitative measures of finding usefulness of Ogmented tool, we measure success rate of participants in each exercise and evaluate relationship between time spent by participants on taking tutorials and their performance in exercises.

4.3.1 Learning Patterns

Exercise 1 involved method binding on click without having participants write code. Evaluation of executing actions by participants is necessary to know for their behavior in learning. As per the logs of Ogmented, learning patterns can be defined as following:

- Active Learning and Training: This kind of learning is observed when participants take actions reflecting upon results which helped them to perform well in exercises. This kind of learning in Ogmented was tracked when users finished doing exactly what they were asked to do. In tutorials, users are taught to create objects and bind methods. Syntax for the same is taught as well. Users are expected to explore other available actions they could perform in tutorials.
- Trial-Error and Random: This kind of behavior is explained when users execute various actions or all possible actions to get expected result. In case of taking tutorials, if users are randomly executing ample of actions in short period of time, we categorized that to trial-error and random category.

4.3.2 Error Types

Exercises include writing of code for following execution:

Creating object of predefined class using "new" keyword using syntax:

ClassName objectName = new ClassName();

Calling method on object created using syntax: *object.method();*

Exercise 2 involved object creation as well as method binding. This exercise shows most errors reported by participants with error rate being 50%. Code can only be executed when statements are syntactically and semantically correct. Hence, we differentiate errors in following categories:

- Syntax errors: Errors caused due to improper use of language syntax. This error occurs in scenarios where there is missing semicolon at end of statement or missing parentheses.
- Semantic errors: Errors caused due to improper use of program statements. An example of cause of this error is when user tries to call a method that is missing declaration.

4.3.3 Error and Success Rate in Exercises

Success rate was calculated by considering total successful attempt and total attempt that participant made in an exercise.

$$\text{Success rate}_e = \frac{successful\ attempts\ in\ exercise\ e}{total\ attempts\ in\ exercise\ e} * 100$$

Error rate = 1- success rate

CHAPTER 5

DATA ANALYSIS AND RESULTS

Data collected from Ogmented logs, pre survey forms, feedback forms and direct observations were used in analysis. Data analysis methods were defined to verify research questions. Analysis of learning pattern and code writing of each participant was explored. Data findings are listed in this section. Methods and metrics defined for analysis are discussed in chapter 4.

5.1 Descriptive Analysis

Pre survey data shows that out of 14 participants, 11 participants had prior experience with programming through Scratch [26]. 3 participants had no experience with coding. 10 out of 14 participants were from 3rd grade while rests were from 4th grade elementary school. Participants were asked to spend nearly 15 minutes with Ogmented. They were really excited to use Ogmented. Objects from Class Zombie were most liked by them. All participants finished all tutorials in average time of 305.642 seconds. 71.428% of participants attempted all exercises. Average time spent on exercises by all participants is 507.5 seconds.

In this chapter, we discuss more about their behavior of learning. We will also discuss about their performance in tutorials and exercises.

5.2.1 Tutorials:

On average, students spent 305.64 seconds on doing tutorials. Tutorials were self-study for participants with instructions to follow presented on the top of display screen. Students were excited to see augmented objects rendered in front of them on marker. We observed that due to this, they sometimes didn't pay as much attention towards instructions. A lot of these participants didn't do well in exercises compared to the ones who spent ample time understanding concepts that each tutorial had to convey to them. Most number of objects were created using Class Zombie. Next frequently used class was Class Car. This showed that Zombie and Car were most famous classes amongst users.

Object name	Count
Zombie 1	84
zombie 2	46
Car1	37
Car2	29
Butterfly1	25

Table 3. Count of objects created by users

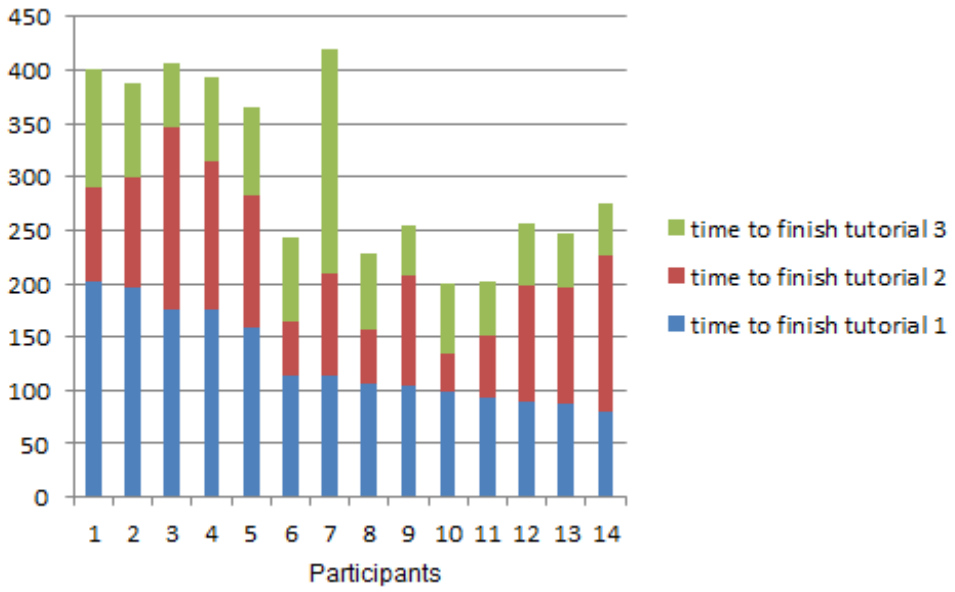


Figure 9. Graph of time taken by users to finish all tutorials

5.2.2 Exercises:

Average of 507.5 seconds was spent by participants to complete 3 exercises. Exercises had range of difficulty levels. Figure 10,11 and 12 show total number of attempts and total successful attempts by all participants. Exercise 1 shows highest number of attempts reflecting random action execution by participant.

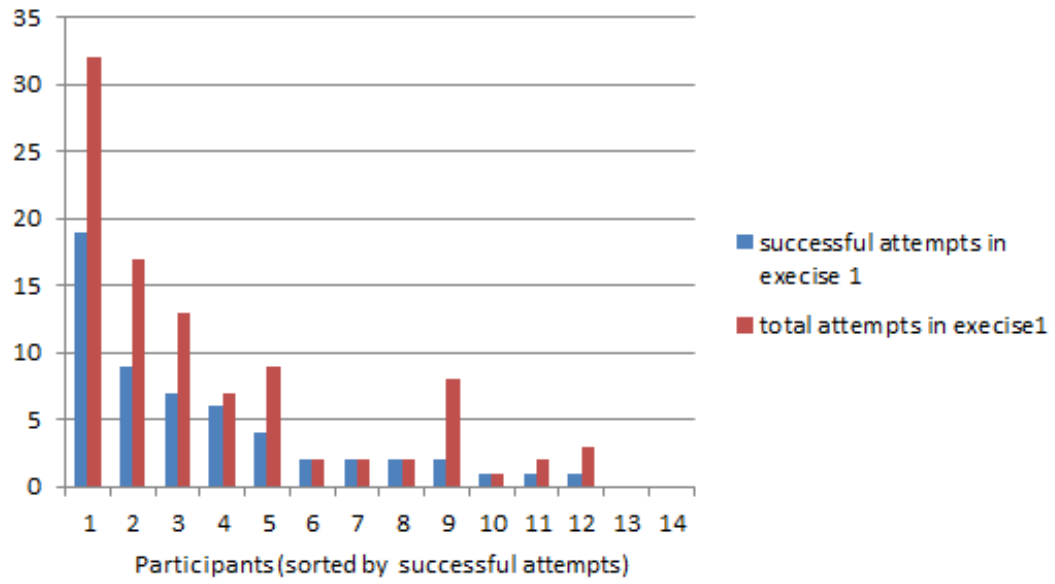


Figure 10. Statistics of exercise 1

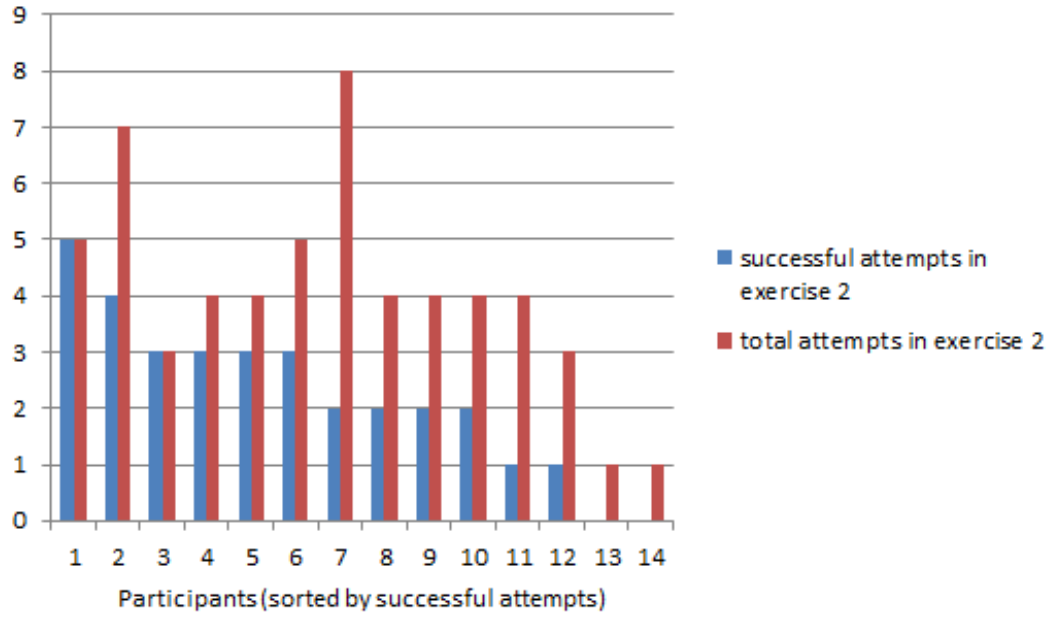


Figure 11. Statistics of exercise 2

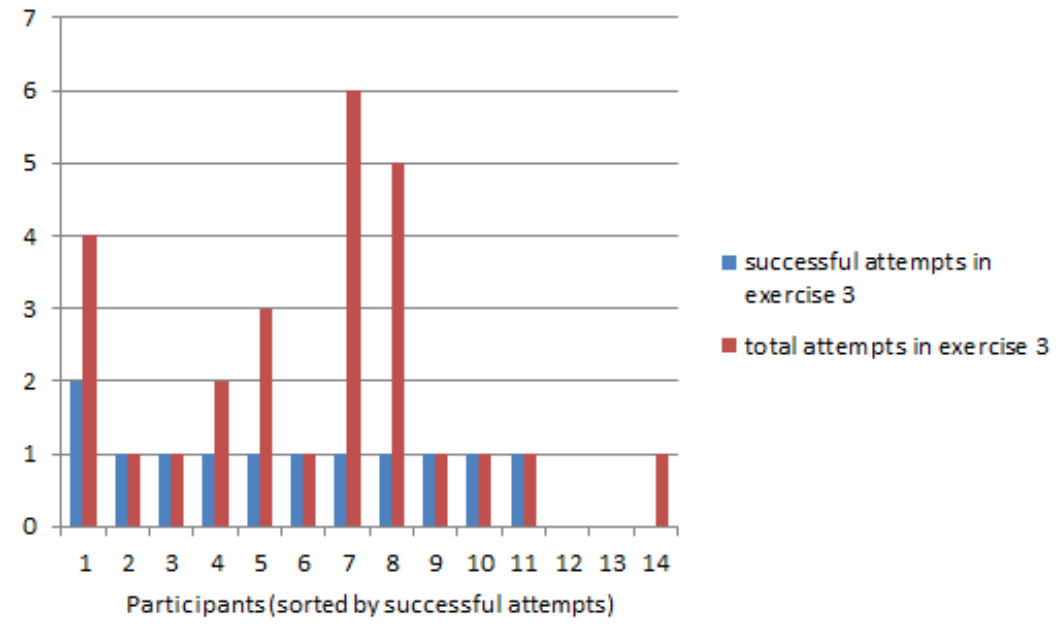


Figure 12. Statistics of exercise 3

5.2 Analysis of Learning Patterns

Purpose of Ogmented is to learn programming concepts using AR. It is thus essential to analyze learning patterns of users which signifies role of Ogmented. To do so, we analyze if participants learnt from tutorial to perform in exercises. Currently to evaluate learning from tutorials, evaluation of amount of time spent and interaction logs is done.

Correlation between time spent on doing tutorials and exercise success rate is 0.072. There is likelihood of having a relation between average time spent on tutorials by participants and their respective exercise success rate. More data is needed for inference of this hypothesis.

We learn about common behavioral pattern by grouping. To do so, plotted points are grouped in 2 clusters using k-means clustering, cluster centroids being 54.744, 131.777 and 53.551, 79.458. Both clusters were examined for common behavior of participants while taking tutorials.

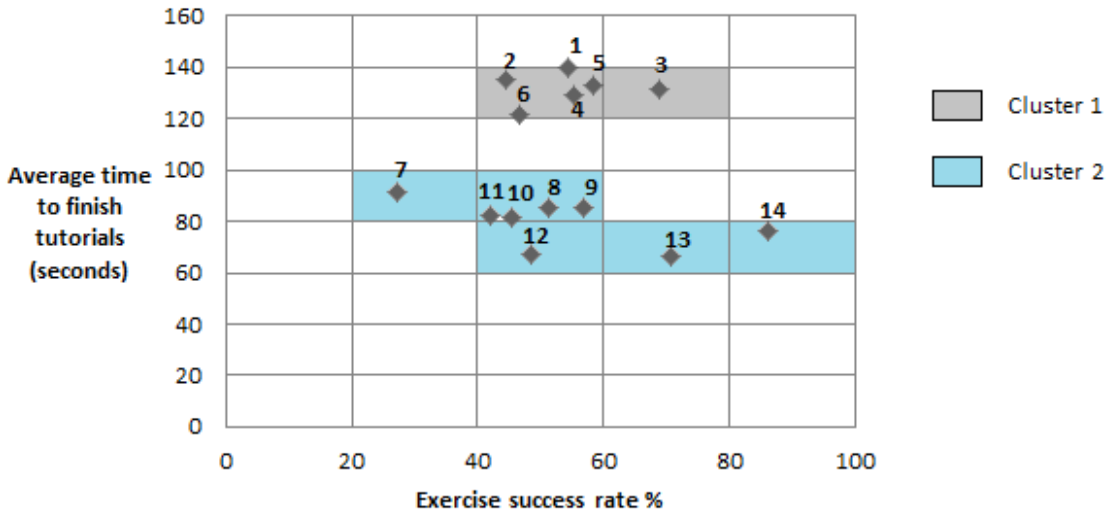


Figure 13. Cluster analysis of performances of participants

Cluster 1 analysis reasons out as to why a participant spending more time in tutorial doesn't learn to perform in exercises.

6 participants that fall under cluster 1 showed following trends:

- Exploration: Most participants in this category showed this behavior. They took same tutorial multiple times. They created objects of all available classes by taking tutorials multiple times as only one/ two objects are allowed to be created in tutorials. This exploratory behavior is important to note. Though they seemed to take longer in tutorials, they did not pay attention to information in information

panel to follow learning. They were distracted from learning and out of curiosity they played around with objects. Zombie class was most popular among them. From direct observation, it is reported that their excitement suppressed learning.

These observations throw light on important design issues. In game based approach of learning, it is very essential that curiosity of exploration of users can be used in a way that leads to enhanced learning [16]. In future work associated with Ogmented, we would incorporate these results to make users focus on learning. Having only limited objects unlocked for tutorials and rest of them locked will make users focus on what is available. Unlocking more objects of different classes can be payoff of finishing tutorial. Chapter 6 speaks more about it.

- Random action execution: Participants who took more amount of time to finish tutorial went on executing actions in random sequence in very less amount of time. It is thus inferred that participants showing this kind of behavior did not actually pay attention to the instructions in information panel on top and hence missed out on learning.

5.2.1 Learning Pattern of Participant 1

This participant spent highest total time (420 seconds) doing tutorials. Upon analyzing his behavior, there are two distinct traits observed that should be discussed.

- This participant took tutorial 2 multiple times. He created 4 objects of class Zombie in tutorial and 2.
- Besides invoking random actions, this participant also showed exploratory behavior. In tutorial 3, methods which were not declared in Butterfly class were invoked by him.

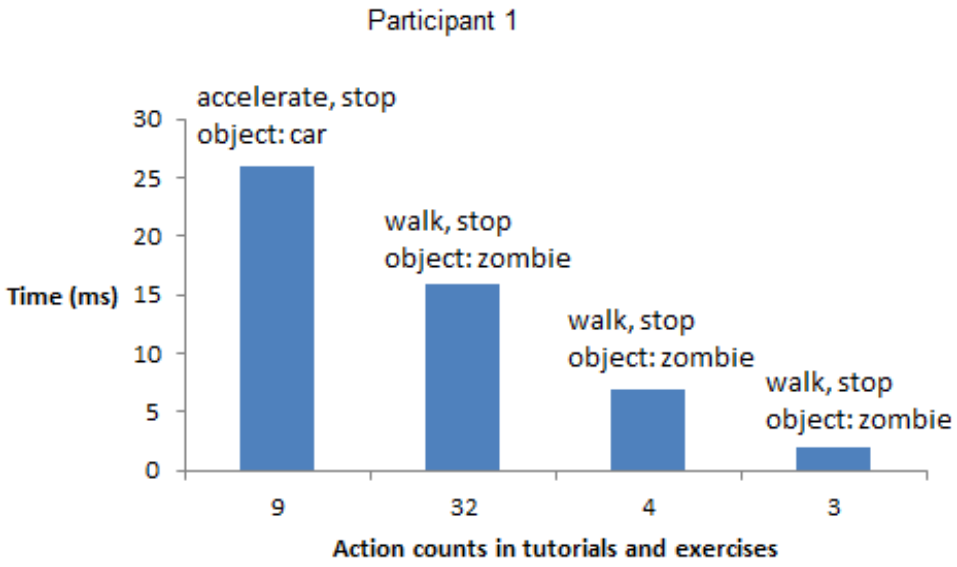


Figure 14. Analysis of actions by participant 1

Figure plots actions that participant executed in small period of time on tutorials and exercise 1. This shows that he finished exercise 1 by trial and error or random execution method.

5.2.2 Learning Pattern of Participant 2

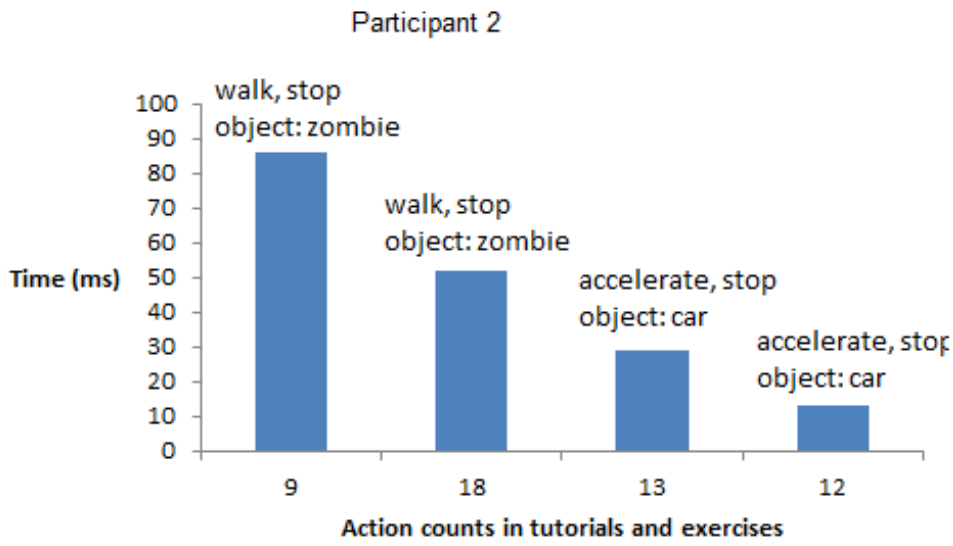


Figure 15. Analysis of actions by participant 2

This participant took second highest total time of 406 seconds to finish tutorials. His learning behavior is very similar to previous participant. Notable observations are:

- Tutorial 2 was taken multiple times; once to create objects from Zombie class and then to do the same for Car class. Zombie class was favorite class by this participant.
- He showed random action execution behavior. Exercise 1 was executed by trial and error approach.
- This participant has likely to be missed out on learning as he engaged with excessive playful interaction with objects from Zombie and Car class.

5.2.3 Learning Pattern of Participant 3

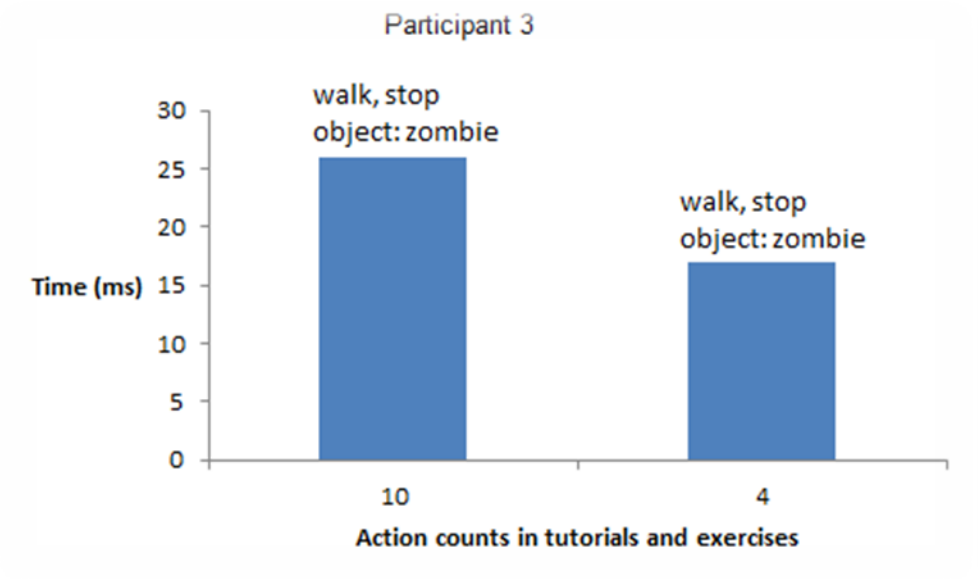


Figure 16. Analysis of actions by participant 3

As previous 2 participants, class Zombie was this participant's favorite class. Difference between behavior of this participant and previous 2 is that this participant did not take tutorial 1 and 2 multiple times. He surely executed random actions though. This participant has higher success rate of 68.98%. There can be a relationship between trying tutorials multiple times to create objects and success rate. Similarly there can also be a correlation between random behavior and success rate. More analysis is needed to infer.

5.2.4 Learning Pattern of Participant 4

This participant has total finish time of tutorials as 387 seconds. Like first and second participants, this participant also takes tutorial multiple times, but to try objects from different classes than a particular class exhibiting exploratory behavior. Semantic errors were made in tutorial by him as methods were invoked that were not declared. He shows curiosity to explore more. As seen in graph, there is less number of repeated action execution patterns.

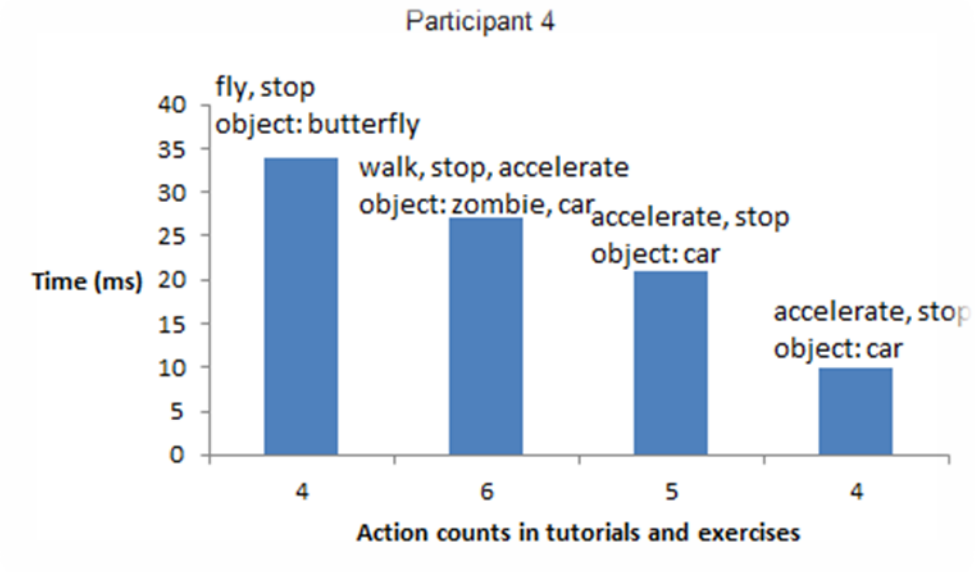


Figure 17. Analysis of actions by participant 4

Graph shows maximum action count as 6. This comes less in random action execution category and more in exploratory learning category. Success rate of exercises is 55.357% for this participant.

5.2.5 Learning Pattern of Participant 5

This participant tried each tutorial at least 3 times to create objects of each of available classes in Ogmented. Tutorial 1 was taken 3 times, tutorial 2 was taken 4 times while tutorial 3 was taken 3 times. In tutorial 3, participants were asked to run pre populated method in code executor. This participant not only tried running prepopulated method fly() but also tried to execute method stop() by himself in tutorial 3. This behavior falls under exploratory behavior. Random execution of actions wasn't seen. This participant took longer simply because trying out tutorials multiple times.

Though calculated success rate is 58.474% for participant 5, there is a unique behavior to observe while exercises were performed by him. He successfully created object car and tried entering code for undeclared method start on car object created in exercise 3. Exercise 3 was not meant to have task of creating car, but as he learned in exercise 2, he gave a shot to car creation to explore. This increased unsuccessful attempt for exercise 3 though he could use his knowledge from previous exercise to successfully create car.

5.2.6 Learning Pattern of Participant 6

This participant took tutorial 2 and 3 multiple times. One of the reasons to get success percentage as 46.735 is that this participant randomly executed 17 actions in exercise 1 in as low as 28 ms. This participant explored all available classes in tutorial 2 while most

interacted object was from class Zombie in tutorials. Other thing to note is that after finishing all exercises, this participant went on to attempt tutorial 1 again.

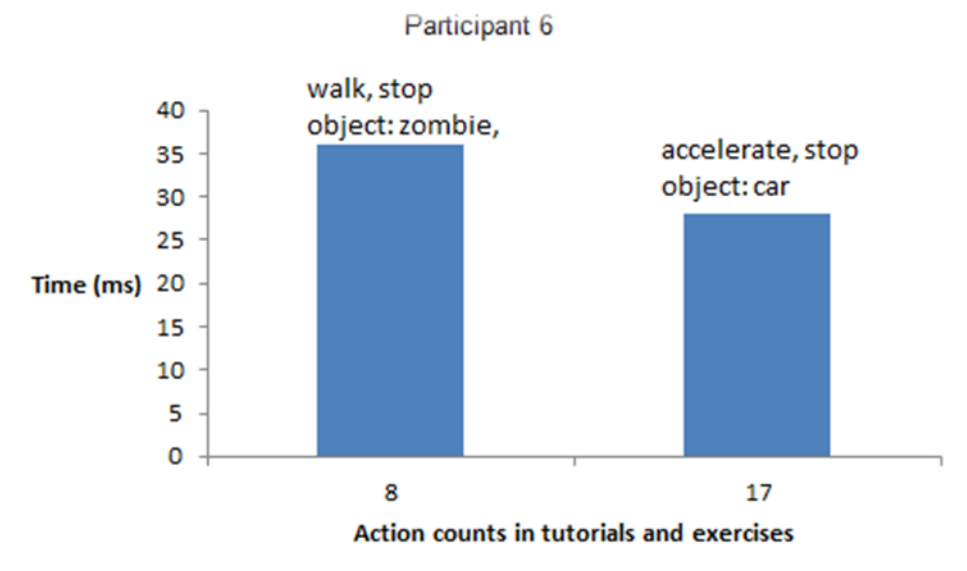


Figure 18. Analysis of actions by participant 6

75% of participants in cluster 2 showed active learning behavior. They finished tutorials by doing exactly what they were asked to do. Thus it seemed that these participants followed the instructions in instruction panel. They reflected upon the results which helped them finish the exercises efficiently. 4 participants in cluster 2 seemed to accidently miss out on an exercise which is therefore essential to have more data to make any kind of inferences. This is also assumed to have been creating some outliers in this cluster.

5.2.7 Learning Pattern of Participant 8

With success rate of 51.271%, total time spent on tutorials by this participant is 255 seconds. This participant did not take any tutorial multiple times. But random trigger behavior was observed for this participant in cluster 2.

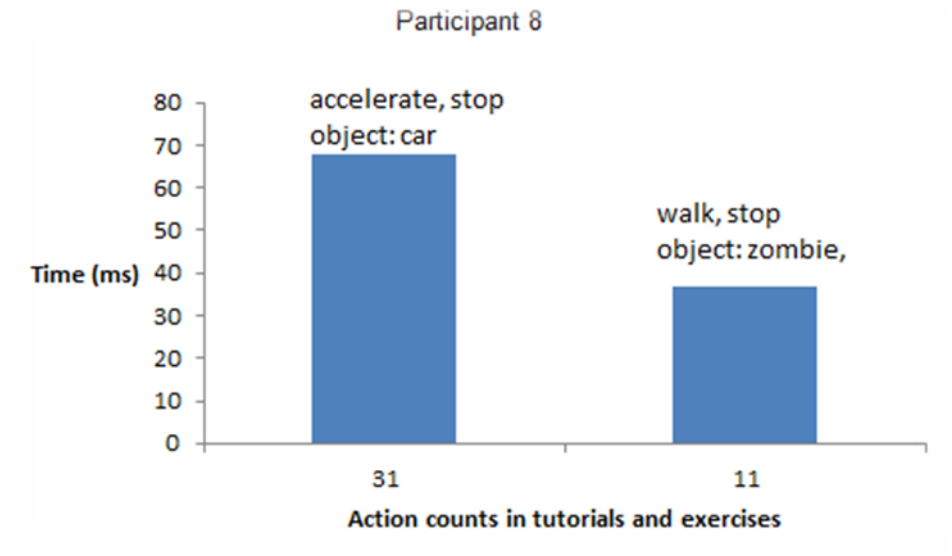


Figure 19. Analysis of actions by participant 8

This random triggering of action was observed to be done in exercise 1, which made success rate go low. Participant did not interact with 2 car objects in tutorials until in exercise 1. This can be a reason of exploration. Thus, irrespective of low time spent on tutorials, this participant might not have better success rate.

5.2.8 Learning Patterns of Participant 9,10,11:

These participants performed exactly what they were asked to perform in tutorial with little of exploration with maximum of one repeated sequence of execution in tutorial. These participants have average performance despite following tutorials. One of the reasons is that each participant either missed out on one of 3 exercises or did not attempt one of the exercises. Participant 11 missed out on 1st exercise; participant 9 missed out on 3rd exercise while participant 10 attempted exercise 2 for 58 seconds and moved to next one. Keeping next exercises locked until current one is finished could eliminate this kind of scenario.

Outliers:

Participants 7, 12, 13 and 14 are located more distantly from the cluster centroid. Their behavior is discussed below:

Participant 7 and 12: Participant 7 took tutorial 2 multiple times to explore all 3 classes in Ogmented. In tutorial 1 and 3, what they were asked to do was covered by them. Participant 12 took all tutorials exactly once and performed what they were instructed to perform. Both participants could not write code using “new” keyword in exercise 2. Just after reading instructions and making one semantic error, they moved to next exercise. In exercise 3 he made one syntax error. This behavior shows that given a chance, participant opted to not perform an exercise.

5.2.9 Learning Pattern of Participant 13

This participant shows active learning behavior with attempting each tutorial once and also with one streak of multiple action execution exploring interactions with object. He also tried to run custom code in tutorial 3 where participants are expected to run prefilled code to get equipped with executing code. This shows exploratory behavior of participant.

5.2.10 Learning Pattern of Participant 14

Behavior of this participant is similar to participant 13. This participant too ran custom code with semantic error in tutorial 3. This explains exploratory behavior. It can be assumed that he followed instructions to complete tutorials.

5.3 Analysis Based on Code Writing

To understand if Ogmented was effective and helpful for participants to teach them to write code snippets, it is essential to analyze code writing pattern. Code writing pattern would also throw some light on participants' ability to implement learning from tutorials. Code snippets entered by users were evaluated. We tracked down where most users made mistakes. Exercise 2 and 3 demanded users to write code, while exercise 1 did not.

Participants had more successful attempts in exercise 1 than rest of the exercises. Table 4 shows success rate in each exercise. Standard deviation of which is 14.373.

Exercise 1	57.475 %
Exercise 2	50.034 %
Exercise 3	55%

Table 4. Average success rates in exercises

Graph below shows number of errors that each participant made in exercises 2 and 3. -1 value is set to represent cases where participants did not attempt to take exercise. Data is sorted with respect to number of errors.

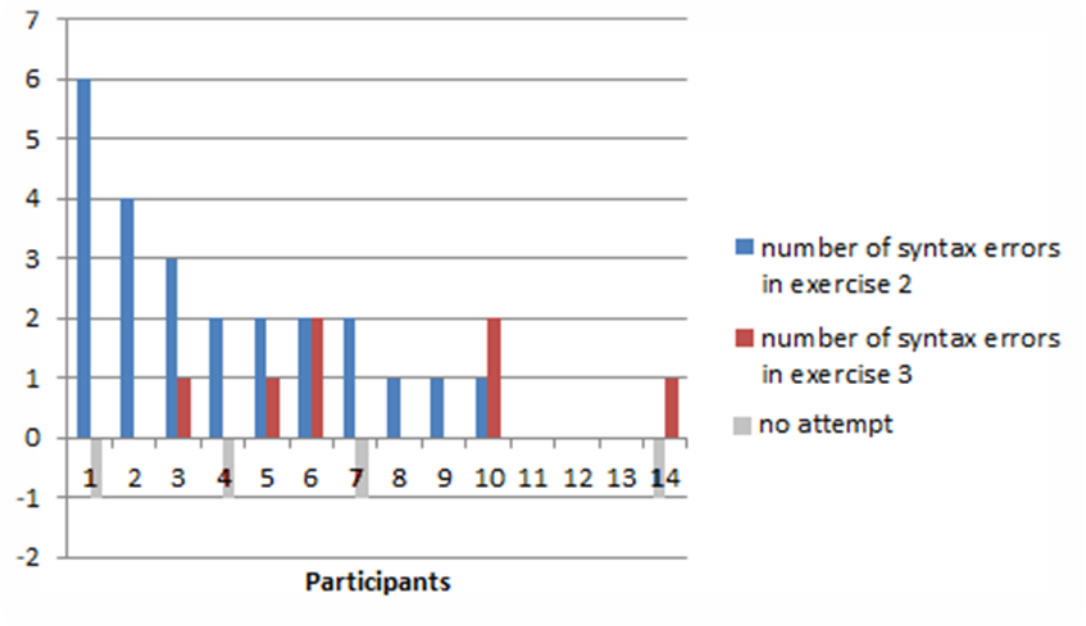


Figure 20. Analysis of errors made by participants

Data shows that most people struggled to write code in exercise 2. It can be inferred that code writing was a challenging task for participants. Out of 14 participants, 6 participants entered code multiple times to create an object using “new” keyword. For method invocation, 10 out of 14 participants did not get syntax right at first. Participants who tried multiple attempts to create object and to call appropriate method in exercise 2, got exercise 3 done in as few as single attempt. Out of 10 people who did not finish exercise 2 in first attempt, 5 of them could finish exercise 3 in first attempt itself.

This shows that exercise 2 was most challenging for majority of participants as it involved writing code to create an object and to call methods. Once participants struggled in getting code right in exercise 2, 71.4% of them could finish exercise 3 successfully with equal or less number of errors. This shows that users evolved from performing in previous exercise to next exercise. Exercise 1 being simplest and not involving tasks to write code has highest success rate.

5.4 Subjective Evaluation

85.7% of all participants successfully managed to execute correct and error free piece of code by end of exercises. 25% of which did not have prior programming experience and 75% of which did not know concepts of object creation, method binding and method invocation. This shows that Ogmented helped as high as 85.7% users to learn to write code for object creation and method binding. 92.857% of participants preferred Ogmented to learn programming. Feedbacks collected after study showed qualitative results of engagement of participants. Some of the feedbacks were as follows:

"They looked good and it was helpful for the codes to know",

"I want to do this a lot more", *"I like it a lot"*.

While some feedbacks written in abstract form by students spoke about their engagement with objects. Those feedbacks are as following:

"Race cars", *"Shoot zombies"*.

Most popular objects amongst participants were from fictitious world objects of class Zombie.

Other important thing to note would be that 21.4% of participants gave up on attempting exercise 3 after their code didn't execute in precious exercise due to syntax errors These participants include participant 2,3 and 4 from cluster 1. All of these participants have likely to not follow instructions on information panel while taking tutorials as they show random action execution in tutorials. Their performance in exercise 1 reflected the same.

CHAPTER 6

CONCLUSION

6.1 Summary

In this work, AR application Ogmented was created for purpose to learn possibility of teaching OOP concepts to novices and elementary students in an interactive way using 3D object rendering in AR. Ogmented modules were designed to first teach concepts in tutorials and later to test users for their learning from tutorials. Each module covered different aspects of object creation and manipulation of object behavior. Computational thinking was implanted by giving real life objects like butterfly, car to interact with. Fictitious world object Zombie was participant's favorite object. Ogmented followed learning using objects and behavior of objects with which users could relate. Benefits or cons of using these objects were not studied in this thesis. But it was assumed to have created engagement in participants.

Logs of exercise performances of users were used to test effectiveness of Ogmented for learning OOP. There was no existing framework for OOP learning using AR rendering. Ogmented introduced possibility of using AR to learn abstract programming concepts.

An user study was conducted with 14 participants from 3rd and 4th class of elementary school. Direct observations reported their enthusiasm to use Ogmented. It showed that students welcomed idea of using AR technology in learning programming. Data analysis of user study revealed following:

- Users who followed instructions in learning syntax and concepts from tutorials demonstrating active learning have likely to be performed well in finishing exercises.
- Learning patterns of participants were categorized into active learning and trial and error based approach of learning. It was observed that game based learning with AR get participants excited and curious. Exploratory behavior logs by participants backed up this observation.
- 85.7% of the participants could write pieces of code proving usefulness of Ogmented. Curiosity in exploring users' favorite objects did distract them from learning. Some of the semantic errors revealed curiosity towards learning though.
- Errors made by users were inspected which showed association with learning. Users, who made more mistakes in exercise 2, bettered themselves in leading exercise. Ogmented showed progressive way of learning. Participants learnt from tutorials and exercises to improve performance in each module.
- Engagement of users was reported qualitatively by direct observations, feedback and quantitatively by calculating how much percentage of users were affected positively and how many users did not give up on finishing all modules in tool. Modules in Ogmented were all optional and not mandatory to move forward.

All the participants showed interest in learning programming and coding in future. This user study shows great insight on using AR to teach abstract programming concepts to novices and elementary students.

While there exists other AR applications to study subjects like Math, Chemistry, Biology, AR can also be used in computer science field for abstract concept learning. This

technology showed potentials to be used in teaching programming to novices and to elementary and high school students.

6.2 Limitations

To prove effectiveness of AR technology against traditional approaches of teaching programming, larger group of participants are needed for any kind of inferences. There was limitation on approaching more number of elementary school students for conducting user study. To measure engagement and learning, users need to be studied for longer term. To evaluate effectiveness of new method, control group vs experienced study needs to be done.

AR technology needs some infrastructure settings. It needs a device to run the application. This device can be a mobile phone or tablet or a computer. Also, there are some head mounted displays that render AR elements superimposed on real world view. Provision of these devices needs some arrangements to be done.

If AR elements are allowed to render anywhere in physical world, learners can get distracted by it. Hence, a lot of AR applications created with purpose of educating learners make use of a marker area in which rendering occurs. More research on this is required to assess rendering method.

6.3 Future Work

Design can be made more usable for users of Ogmented. In future, this application can be enhanced to not follow “one size fits all” concept. Machine learning techniques can be used to provide exercises to users based on their progress. Difficulty level of modules can be adjusted by internal algorithms. There can be an enhanced version of Ogmented for more efficiency.

Also, results from data analysis showed that having all objects unlocked for participants distracted them from intended learning sessions. In improvised version of Ogmented, this issue is expected to be taken care of. Payoff of unlocking treasures upon finishing tasks is set up high on priority for implementation.

Besides, an user study with more number of participants can be conducted to measure effectiveness of AR technology to learn programming. More comprehensive analysis would throw light on broader aspects.

REFERENCES

[1] Paulo Blikstein. Using learning analytics to assess students' behavior in open-ended programming tasks. Proceeding LAK '11. Proceedings of the 1st International Conference on Learning Analytics and Knowledge Pages 110-116.

[2] Kelleher, C. and Pausch, R. Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. ACM Computing Surveys 37, 2 (June 2005), 83–137.

[3] Suzanne Stokes. Visual Literacy in Teaching and Learning: A Literature Perspective. Electronic Journal for the Integration of Technology in Education, vol. 1, no. 1

[4] Lori Carter. Why students with an apparent aptitude for computer science don't choose to major in computer science, Proceeding SIGCSE '06, Pages 27-31.

[5] Tony J. A Participative Approach to Teaching Programming, ACM SIGCSE, Volume 30 Issue 3, Sept. 1998, Pages 125-129 1998.

[6] Mark J. Graham, Jennifer Frederick, Angela Byars-Winston, Anne-Barrie Hunter, Jo Handelsman. Increasing Persistence of College Students in STEM, Science 27 Sep 2013:
Vol. 341, Issue 6153, pp. 1455-1456

[7] William E Senior Dugger. Evolution of STEM in the United States

[8] M. McCracken, V. Almstrum, D. Diaz, M. G. anD. ianne Hagan, Y. B. Kolikant, C. Laxer, L. Thomas, I. Utting, and T. Wilusz. A multi-national, multi-institutional study of assessment of programmingskills of first-year CS students. SIGCSE Bulletin, 33(4):125–180, 2001.

[9] R. Lister, O. Sepp¨al¨a, B. Simon, L. Thomas E. S. Adams, S. Fitzgerald, W. Fone, J. Hamer, M. Lindholm, R. McCartney, J. E. Mostr¨om, and K. Sanders. A multi-national study of reading and tracing skills in novice programme rs. SIGCSE Bulletin, 36(4):119–150, 2004

[10] J. Tenenberg and S. F. et al. Students designing software: a multi-national, multi-institutional study. Informatics in Education, 4(1):143–162, 2005

[11] Fleury A. Acting Out Algorithms: How and Why It Works. Proceedings of the 4th Annual CCSC Midwestern Conference, Dominican University, September 1997.

[12] Elliot Soloway. Learning to Program = Learning to Construct Mechanisms and Explanation,1986

[13] Graham, M., Zook, M., and Boulton, A. "Augmented reality in urban places: contested content and the duplicity of code." Transactions of the Institute of British Geographers, DOI: 10.1111/j.1475-5661.2012.00539.x 2012.

[14] Tamra Ogletree. Mix Methods Research "Findings Report" Letters Alive: Case Study, Alive studios

[15] Spielberger, Starr et al. Curiosity and Exploratory Behavior. (1994). Motivation: Theory and research, (pp. 221-243). Hillsdale, NJ, US: Lawrence Erlbaum Associates, Inc, viii, 332 pp.

[16] Vilorio D. STEM 101: Intro to tomorrow's jobs. Occupational Outlook Quarterly. 2014;58(1):2-12.

[17] Wilson C, Sudol LA, Stephenson C, Stehlik M. Running on empty: The failure to teach K-12 computer science in the digital age. Association for Computing Machinery. 2010 (p. 26)

[18] Malone, T. (1980). What makes things fun to learn? Heuristics for designing instructional computer games. Proceedings of the 3rd ACM SIGSMALL Symposium and the 1st SIGPC Symposium (pp. 162–169). Palo Alto, USA.

[19] Kelleher, C. and Pausch, R. Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. ACM Computing Surveys 37, 2 (June 2005), 83–137.

[20] David J. Malan, Henry H. Leitner. Scratch for Budding Computer Scientists. SIGCSE'07, March 7–10, 2007, Covington, Kentucky, USA. Copyright 2007 ACM 1-59593-361-1/07/0003

[21] M. Moskal, D. Lurie, and S. Cooper, Evaluating the Effectiveness of a New Instructional Approach. Conference '00, Month 1-2 2000 ACM 1-58113-000-0/00/0000

[22] Resnick, M., Maloney, J., Monroy-Hernandez, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., & Kafai, Y. (2009). Scratch: Programming for All. Communications of the ACM, vol. 52, no. 11, pp. 60-67 (Nov. 2009)

[23] Cagin K, Mary K, Liz Bacon, Lachlan Mackinnon et al. A Serious Game for Developing Computational Thinking and Learning Introductory Computer Programming. Procedia - Social and Behavioral Sciences. Volume 47, 2012, Pages 1991-1999

[24] Sengupta, P., Kinnebrew, J.S., Basu, S. et al. Integrating Computational Thinking with K-12 Science Education. Educ Inf Technol (2013) 18: 351. doi:10.1007/s10639-012-9240-x

[25] Sze Yee Lye, Joyce Hwee Ling Koh. Review on teaching and learning of computational thinking through programming: What is next for K-12? Computers in Human Behavior, Volume 41, December 2014, Pages 51–61

[26] https://scratch.mit.edu/
Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., ... & Kafai, Y. (2009). Scratch: programming for all. Communications of the ACM, 52(11), 60-67.

[27] http://www.alice.org/index.php
Kelleher, C. and Pausch, R. Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. ACM Computing Surveys 37, 2 (June 2005), 83–137.

[28] http://www.augment.com/
This Augmented Reality app could change the way we shop online, Mashable, 2013-03-11

[29] http://elements4d.daqri.com - Created by DAQRI, Los Angeles, CA
Krajcik, J.S. & Mun, K. (2014). Promises and challenges of using learning technology to promote student learning of science. N. Lederman & S. Abell (eds.)Handbook of Research in Science Education, Vol II, p. 337-360

[30] https://www.aurasma.com/
Becky Sue Parton, Robert Hancock. Animating the Inanimate using Aurasma: Applications for Deaf students. Society for Information Technology & Teacher Education International Conference, Mar 05, 2012 in Austin, Texas, USA ISBN 978-1-880094-92-1