

SANTA CLARA UNIVERSITY
Department of Computer Science and Engineering

Date: June 7, 2022

I HEREBY RECOMMEND THAT THE THESIS PREPARED
UNDER MY SUPERVISION BY

Shivangi Kar
Stephanie Lu

ENTITLED

ORIENT: Teaching Object Oriented Programming with Augmented Reality

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

BACHELOR OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING



Ihan Hsiao
06/08/2022

Thesis Advisor



N. Ling (Jun 8, 2022 11:35 PDT)

Department Chair

ORIENT: Teaching Object Oriented Programming with Augmented Reality

By

Shivangi Kar

Stephanie Lu

SENIOR DESIGN PROJECT REPORT

Submitted to
the Department of Computer Science and Engineering
of

SANTA CLARA UNIVERSITY

in Partial Fulfillment of the Requirements
for the degree of
Bachelor of Science in Computer Science and Engineering

Santa Clara, California

June 7, 2022

ORIENT: Teaching Object Oriented Programming with Augmented Reality

Shivangi Kar

Stephanie Lu

Department of Computer Science and Engineering

Santa Clara University

June 7, 2022

ABSTRACT

In our technologically advanced society, computational thinking is a critical skill for students to develop. Our project, ORIENT, is a mobile application that uses augmented reality to teach object-oriented programming (OOP), a fundamental concept in computer science. ORIENT is designed for novice programmers from the middle school level up, and it consists of a three-part tutorial series that teaches class creation, inheritance, and polymorphism—three of the most important tenets of object-oriented programming. ORIENT, which was built in Unity and made specifically for the iPad, includes a series of interactive tutorials with immediate feedback; it encourages students to explore OOP in an guided environment that prioritizes learning over gamification. Students can learn at their own pace and easily navigate between and within tutorials. Through user testing with novice programmers in middle school, high school, and college, we found ORIENT to be highly effective in communicating OOP concepts in an engaging and understandable manner. We hope that ORIENT can provide insight on how immersive technology can be used to enhance education, particularly in the field of computer science.

ACKNOWLEDGEMENTS

We would like to thank our advisor, Dr. Sharon Hsiao, for her guidance throughout this project. Additionally, we would like to thank the School of Engineering and the Computer Science and Engineering department in particular for their support and recognition of our work.

Table of Contents

1	Introduction	1
1.1	Problem Statement	1
1.2	Motivations	1
1.3	Related Work	2
1.3.1	Alice	3
1.3.2	OOP-AR	4
1.3.3	Ogmented	5
1.4	Objectives	6
2	Proposed Solution	8
2.1	Design Rationale	8
2.2	Tutorial Content	9
2.2.1	Tutorial 1: Class Creation	9
2.2.2	Tutorial 2: Inheritance	12
2.2.3	Tutorial 3: Polymorphism	16
3	Technologies Used	20
3.1	Unity.....	20
3.2	C#	20
3.3	ARKit	20
3.4	AR Foundation	20
4	Performance Evaluation	22
4.1	Methodology	22
4.2	Empirical Results	22
4.3	User Feedback	24
5	Future Work	26
6	Societal Considerations	27
6.1	Ethical Justification	27
6.2	Privacy	28
6.3	Quality of Education	28
7	Conclusion	30

List of Figures

1.1	Alice Programming Interface	3
1.2	Visuals in Alice	4
1.3	OOP-AR User Interface	4
1.4	Method Binding in Ogmented	6
2.1	Tutorial 1 Attributes Exercise	10
2.2	Tutorial 1 Methods Exercise	11
2.3	Tutorial 1 Car Instantiation	11
2.4	Tutorial 2 Inheritance Lesson	12
2.5	Tutorial 2 Car and PoliceCar Relationship	13
2.6	Tutorial 2 PoliceCar Declaration	14
2.7	Tutorial 2 PoliceCar Attribute Selection	15
2.8	Tutorial 2 PoliceCar Instantiation	15
2.9	Tutorial 2 Inheritance Tree Diagram	16
2.10	Tutorial 3 Polymorphism Lesson	17
2.11	Tutorial 3 Applying Polymorphism	18
2.12	Tutorial 3 Method Overriding	18
2.13	Tutorial 3 PoliceCar and Car AR Exploration	19

Chapter 1

Introduction

1.1 Problem Statement

In our technologically advanced society, it is critical for students to develop an understanding of computer science fundamentals as well as computational thinking skills. The existing availability of computer science education does not meet demand; a Gallup poll found that although 90% of parents would like their children to learn the subject, only 47% of high schools offer computer science courses, and the subject is absent in most middle and elementary schools [1]. Therefore, there is a need for accessible educational content that teaches computer science basics to young students.

One important computer science concept in particular is object-oriented programming (OOP), which is fundamental to programming languages like Java, C++, and Python. OOP is a crucial topic because it enables programmers to write more flexible, concise, and reusable code. However, understanding the relationship between parent and child classes, their interactions, and other OOP principles has proven difficult for many students. A significant instructional challenge is that OOP is more abstract than structured programming and more demanding in the analysis and design processes. Thus, a visual and hands-on approach may be more effective in teaching these programming principles.

Our senior design project investigates novel ways to utilize immersive technology in aiding computer science education. Specifically, we explore the effectiveness of using augmented reality to help students visualize abstract concepts in object-oriented programming, focusing primarily on the principles of class creation, inheritance, and polymorphism.

1.2 Motivations

Our decision to utilize augmented reality (AR) for our project is rooted in the widespread availability of AR, as well as the demonstrated effectiveness of AR as an educational tool.

Augmented reality is the modification of real-world environments through the incorporation of computer-generated perceptual information. Technology is used to add visuals, sounds, haptic feedback, and other sensory inputs to enhance a user’s experience of reality [2]. AR experiences are available through most modern mobile devices, making AR increasingly accessible and popular. Headsets—which can be cost-prohibitive—are not required to experience AR, making AR more accessible than virtual reality.

As education becomes increasingly digitized, mobile augmented reality has emerged as a learning tool. Its popularity is driven by two primary factors [3]. First, smartphones and tablets have become widespread and advanced, with high resolution cameras, GPS sensors, and vibrant displays. Educational institutions and students often already possess the mobile devices they need to use AR. Second, the availability of AR software development kits has enabled developers like ourselves to easily create custom AR apps. AR applications that explore a variety of subjects are constantly being developed and released on app stores, providing plenty of options for students and educators.

The effectiveness of AR as an educational tool has been demonstrated by numerous studies. AR can help facilitate learning because it enables exploration and visualization in 3D space. When compared to students who do not use AR to learn a particular topic, students who do use AR show increased understanding, intrinsic motivation, engagement, retention, and performance [4]. As such, we are motivated to apply AR toward computer science education, such that novice programmers may understand object-oriented programming concepts more easily and in a more engaging manner.

1.3 Related Work

Currently, novice programmers can learn to program using popular applications like Scratch and Kodable. Scratch is a block-based visual programming language and web app that enables early coders to build interactive games and animations [4]. However, it is not an object-oriented programming language and is not suitable for teaching OOP principles [5]. Kodable is an app aimed at elementary school students that also teaches computer science concepts through visuals and interactive games [6]. While it does utilize the OOP language JavaScript, it is restricted to four very basic topics: classes, properties, functions, and subclasses. There are no existing applications available to the public that teach object-oriented programming using AR technology, which has proven effective in helping students understand biology, math, and other subjects.

Alice, OOP-AR, and Ogmented are the closest existing solutions with comparable objectives for teaching object-oriented programming.

1.3.1 Alice

Alice is a visual based programming language commonly used in introductory computer science classes. Students are presented with a list of objects, attributes, and logic blocks they can drag and drop from Alice's gallery into their virtual environment. They are able to “program” functionality by placing these blocks together in the appropriate order.

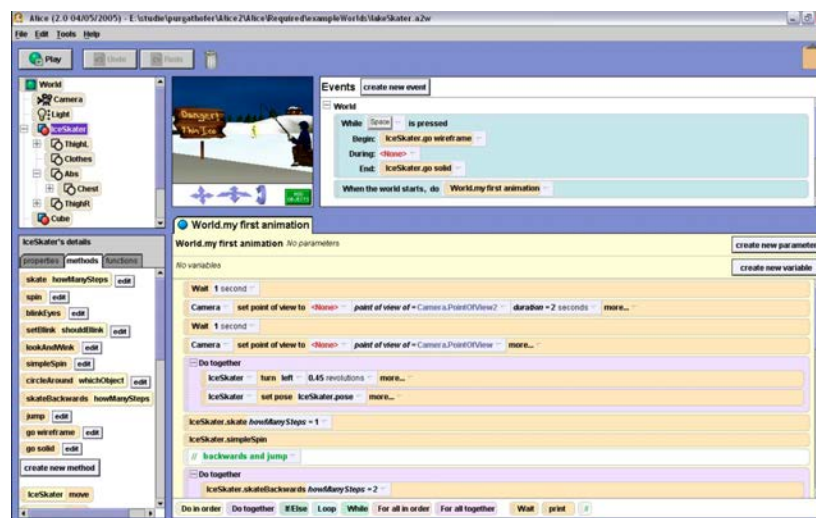


Figure 1.1: Alice programming interface with drag-and-drop logic blocks [7]

While Alice is a great tool, it is not a guided learning platform. The drag and drop without specific prompts and instructions can be confusing and overwhelming for individuals who do not fully understand what the logic or attribute blocks entail. Additionally, the visuals are limited to one's computer screen. The latest version of Alice supports virtual reality integration, but because this requires additional tools, it is not accessible to all students [8].



Figure 1.2: Visuals in Alice do not incorporate the user's real-world environment [9]

1.3.2 OOP-AR

OOP-AR is an Android application with similar goals of teaching OOP concepts using augmented reality in classroom settings. Designed by researchers at Universiti Pendidikan Sultan Idris in Malaysia, it is intended for use by university-level students [10].

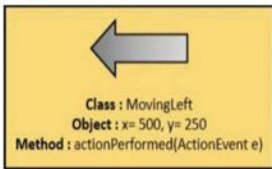

Left Movement		
Target Marker	Output	
		
Procedures	Expected Output	Result
Scan the target marker in AR Camera to get the output	The output should be a 3D butterfly flying to the left.	Failed (The butterfly only faced to left and not flying)

Figure 1.3: OOP-AR marker card and corresponding functionality [10]

In the diagram above, the left side depicts a marker, and the right side shows how the marker is read to generate an output in AR. In this example, the target marker includes three fields: Class, Object and Method. The Class attribute describes the direction the object should move. The object's original position is represented by the Object attribute. The Method attribute is simply an action event handler. Overall, the app includes 5 target markers for different directions such as up down left right and static. If the marker card works as intended, when the user scans it, a 3D representation of a butterfly should appear and travel in the direction indicated on the card.

However, there are some clear drawbacks with OOP-AR. First, it does not fully work as intended. When the user scans the target marker in the diagram above, the expected output is a 3D butterfly model moving left. However, the program fails. The butterfly object is generated and faces the specified direction but does not move. Additionally, this app was designed for novice programmers to learn object oriented programming. However, it is restricted to just object creation, where the user has little understanding of the various attributes or actions that an object can have or perform. Despite the fact that the marker uses key terms such as Class, Object, and Method, there is little description or explanation of what these terms mean and how they contribute to the output 3D model. Furthermore, it does not cover other important OOP concepts like inheritance or polymorphism.

1.3.3 Ogmented

Ogmented is an Android app that uses augmented reality to teach students abstract programming topics. It explores the pedagogical effects of utilizing real-world 3D models for learning. The two main sections of Ogmented are tutorials and exercises. Users must first complete tutorials to become acquainted with the tool, coding concepts, and syntax. Through 3D rendering and visual programming, tutorials are designed to help students learn programming concepts and syntax. After completing the tutorials, users can attempt exercises designed to put their tutorial knowledge to the test [11].

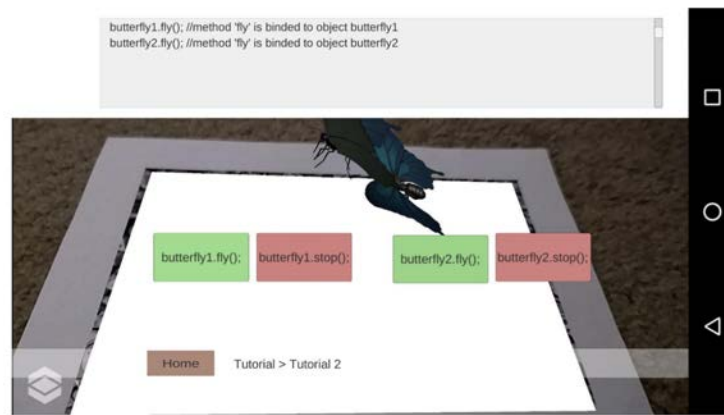


Figure 1.4: Tutorial 2 Method Binding in Ogmented [11]

The primary focus of Ogmented is object creation, method binding and method invocation. A drawback of the app is that it does not extend to any other tenet of OOP. All classes are also predefined, so users are not required to think critically about the different fields and methods associated with an object. They are simply given methods to call and instructed to click on buttons to see how they work. It is critical that novice programmers understand that objects behave the way they do because of the code written for the object; this connection is not made clear in Ogmented.

1.4 Objectives

Taking into consideration the drawbacks of existing applications, our app ORIENT was designed with three key objectives in mind.

Objective 1

The first objective is to present educational content in an age-appropriate manner. This entails making the content easy to understand and UI easy to follow. Students are introduced to key programming terms such as class, object, polymorphism, etc. through smaller exercises, making the material digestible and not overwhelming. Through the use of visual components, the application is intended to make the learning process and AR experience as simple as possible.

Objective 2

Viewing a large body of code can often be intimidating for beginner programmers, causing frustration and distracting students from the learning process. Our second objective is to build an intuitive AR experience where students are not bogged down with learning vocabulary or code syntax. This entails creating exercises where a user's interactions have clear, corresponding impacts on an AR visualization. Students are able to follow structured tutorials in a guided learning environment that decomposes and visualizes complex code.

Objective 3

Our final objective is to encourage students to explore OOP with learning as the primary focus. Technology, specifically augmented reality, can be distracting. While seeing an object appear in one's real world environment is exciting, the addition of such virtual features can cause students to become engrossed in a game and addicted to it. This may lead to information overload and overreliance on technology. Instead of perceiving an educational AR experience as an addicting game, students' primary focus should be on learning. Thus, our application is designed to avoid taking the user's attention away from important environmental or real-world cues. ORIENT encourages its users to keep learning in a healthy, non-distracting way by providing digestible content and a positive feedback cycle.

Chapter 2

Proposed Solution

ORIENT is a mobile app created in Unity and designed specifically for the iPad. It consists of a series of interactive tutorials that utilize AR to teach object-oriented programming concepts. The app encourages students to explore OOP in a guided environment that prioritizes learning over gamification. ORIENT's three-part tutorial series is aimed at teaching novice programmers about the core OOP concepts of class creation, inheritance, and polymorphism. Unlike existing solutions, our platform provides a guided learning environment with immediate feedback and clear learning outcomes. While playing around with AR objects and interacting with the tutorial exercises, students are introduced to actual code syntax and structure. ORIENT is designed to encourage students to learn at their own pace by providing simple content and an easy-to-use interface.

2.1 Design Rationale

In the current educational technology market, there are a limited number of applications with the core objective of introducing OOP concepts to beginning programmers using AR. Unfortunately, existing solutions do not cover all aspects of OOP, have missing features, or require a steep learning curve to become familiar with the software.

When we first started developing ORIENT, we had to decide whether the app should be more entertaining and engaging, with self-guided augmented reality exploration, or more structured and focused on predefined learning exercises. While the idea was to capitalize on the novelty of augmented reality by allowing students to experiment freely with it, we also had to ensure that students satisfied certain learning objectives. We observed that many of the current solutions adopted a more structured approach, because feedback and learning were easier to impart with defined instructions. By introducing augmented reality into ORIENT, we were able to provide a

sense of play and self-directed learning. Thus, ORIENT incorporates both structured tutorials that also enable users to experiment and interact with the AR models they create at the end of each tutorial.

Furthermore, while inheritance and polymorphism are fundamental aspects of OOP, they are often overlooked in existing applications. In object-oriented programming, inheritance promotes code reusability and reduces code length. Polymorphism allows the object to choose which form of the function to implement both at compile (overloading) and run time (overriding). For novice programmers, understanding how inheritance and polymorphism work is critical since it can help them establish better programming habits and generate more concise and efficient code in the long run. As a result, ORIENT dedicates two of its tutorials to breaking down these broad concepts into manageable chunks. Each tutorial builds on the previous, allowing users to understand both what goes into a class or object and how it interacts with other classes and corresponding objects.

Additionally, since ORIENT is designed for younger audiences (middle school level and up), it centers its tutorials and exercises around a Car class and object. Because automobiles are the most common mode of transportation in the United States, it is reasonable to assume that the majority of users are familiar with the various characteristics and actions of a car. ORIENT educates through an object that users are already familiar with.

2.2 Tutorial Content

Orient is divided into three tutorials, each of which builds upon the previous tutorial. The first tutorial focuses on class creation, the second on inheritance, and the third on polymorphism.

2.2.1 Tutorial 1: Class Creation

Tutorial 1 introduces students to the concept of class creation. The goal of this lesson is to create a Car class and instantiate a Car object. There are two primary interactive exercises that help

students understand that the characteristics of a car—such as color, make, and model—are attributes, and that a car’s behaviors—such as driving, turning, or honking—are methods.

The first target outcome for this exercise is to declare and define a Car class. The first exercise asks users to determine which attributes are applicable to a car.

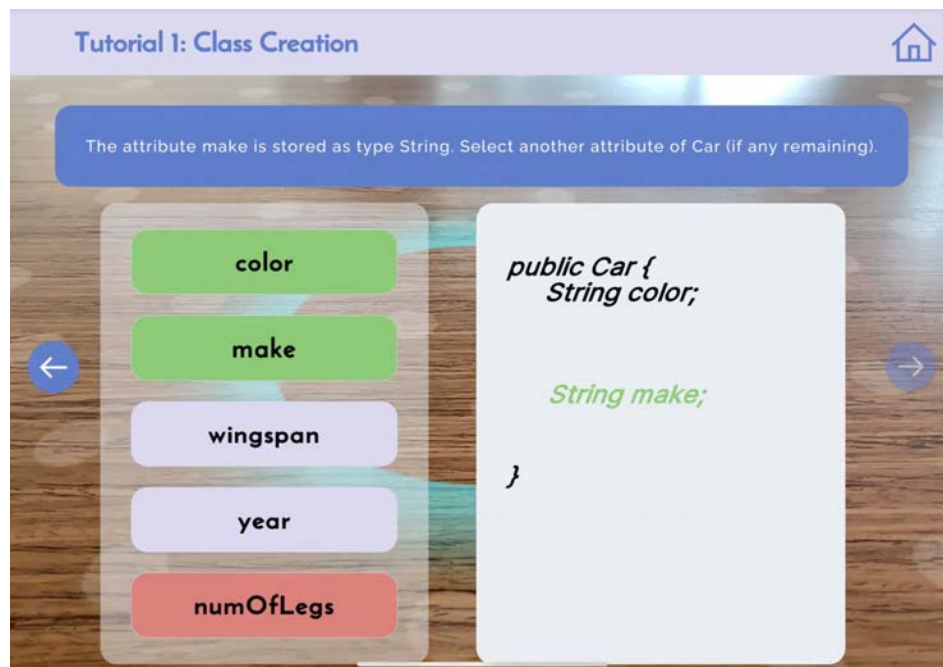


Figure 2.1: Tutorial 1 Attributes Exercise

In the diagram above, the left-hand side lists a number of attributes and the right-hand side is a declaration of the Car class. Users are prompted to select specific Car attributes or methods, which populate the code for the Car class on the right panel. If the user’s selection is correct, they will be given a detailed explanation of how it translates to code. For example, if the user selects "color" as an attribute of Car, a popup will appear explaining that color represented in code as a string. Similarly, if the user selects an incorrect characteristic, they will be redirected after receiving a brief explanation for why they chose the incorrect response.

This exercise is replicated with specific methods as well. Users are again asked to choose methods they believe are relevant to a car, and the process is repeated until all correct methods have been chosen.



Figure 2.2: Tutorial 1 Methods Exercise

The second target outcome is to instantiate a Car object in AR. To do this, the user will type code to create a Car object, and that object will appear in front of them in their own environment. The user can then interact with the Car object in augmented reality by clicking on the purple buttons to see it driving, turning, reversing, or honking.

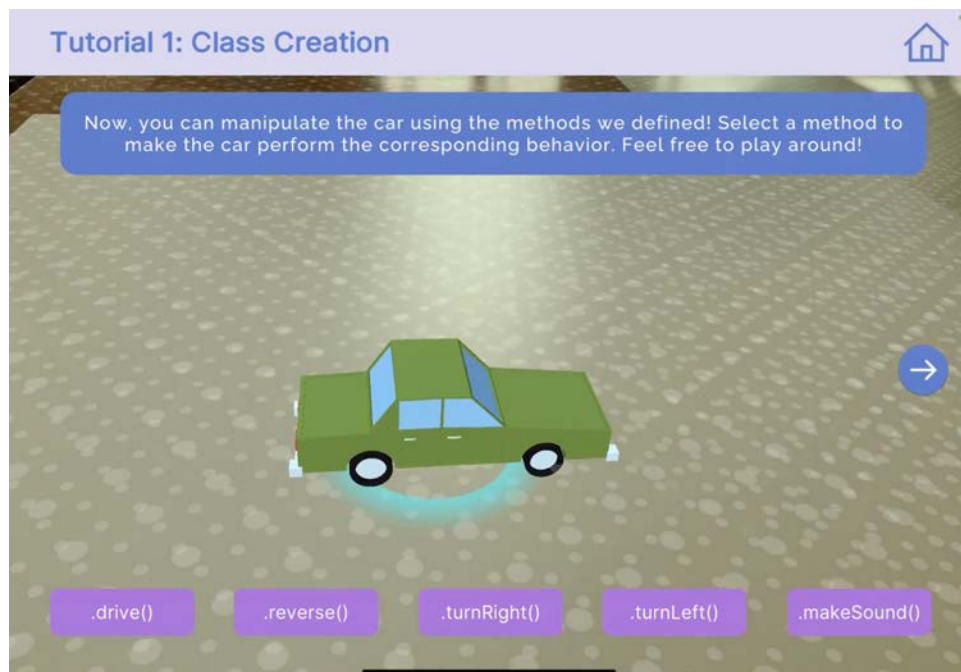


Figure 2.3: Tutorial 1 Car Instantiation

2.2.2 Tutorial 2: Inheritance

The goal for Tutorial 2 is to teach students how to distinguish between the inherited and unique attributes and behaviors of a subclass.

By the end of this tutorial, users will be able to:

1. Recognize that the PoliceCar class can use and apply the fields and methods of the Car class.
2. Define attributes and behaviors unique to a police car.
3. Instantiate a PoliceCar in AR to test out inherited methods and methods unique to the class.

The first learning outcome aims to guide users through creating the PoliceCar built upon the existing Car class from Tutorial 1. The notion of inheritance is first introduced to users through a step-by-step explanation that corresponds to the car and police car examples used throughout the application. This way, the student is making connections with information they are already familiar with and worked with in Tutorial 1.

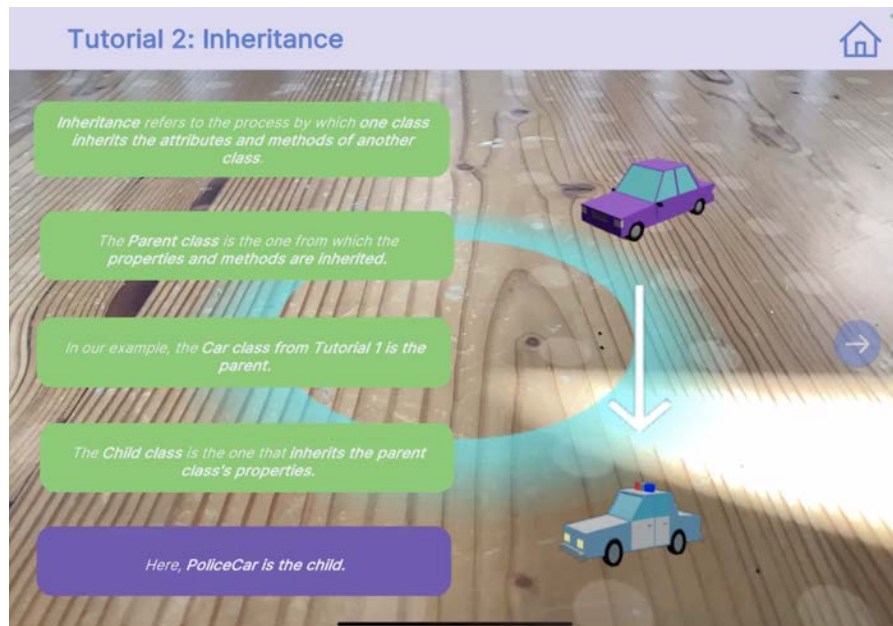


Figure 2.4: Tutorial 2 Inheritance Explanation

Students may observe how properties like year, color, and model can be taken from one object and applied to another. Specifically, the Car object can be used to make other objects like police cars, trucks, buses, and vans due to their similarities. Thus, because a police car is a type of car, all of the attributes and behaviors that apply to a car can also apply to a police car. In the following exercise, students are able to see how a parent and child class relationship translates to code.

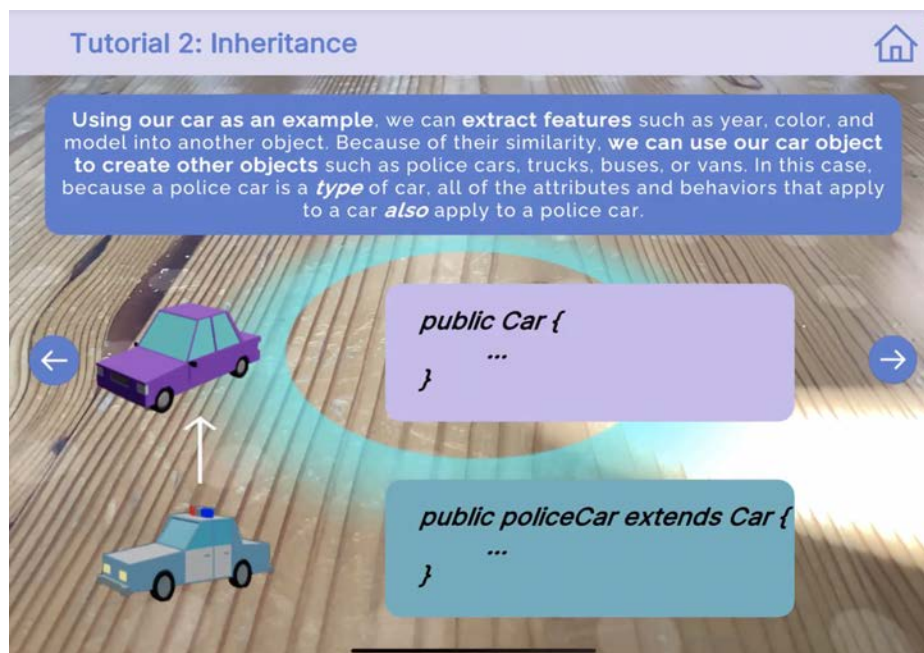


Figure 2.5: Tutorial 2 Car and PoliceCar Relationship

Students are then asked to refer to this exercise and determine how to declare the PoliceCar class in Figure 2.6. If users make a mistake, a pop-up feedback box prompts them to reconsider the class being inherited in this scenario.

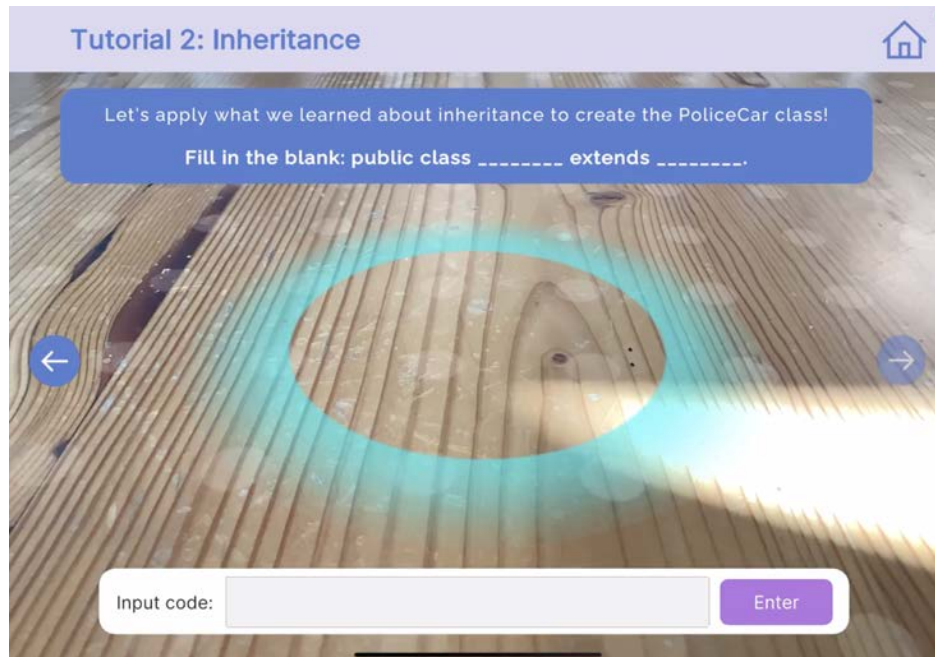


Figure 2.6: Tutorial 2 PoliceCar Declaration

After declaring the PoliceCar class, students are asked to look at some of the distinct features of a police car. This leads to the second objective, which is to help the user understand that a PoliceCar is a subclass of Car. This means that it inherits certain fields and methods, but also has its own unique attributes. Similar to Tutorial 1, the next exercise asks the user to select attributes they believe correspond to a police car (Figure 2.7). If they select the correct attribute, the corresponding code appears on the right hand side of their screen. The green feedback boxes let them know they are correct and provide insight as to why and how this can be represented in code. The red feedback boxes mean they were incorrect, but provide encouragement to try again.

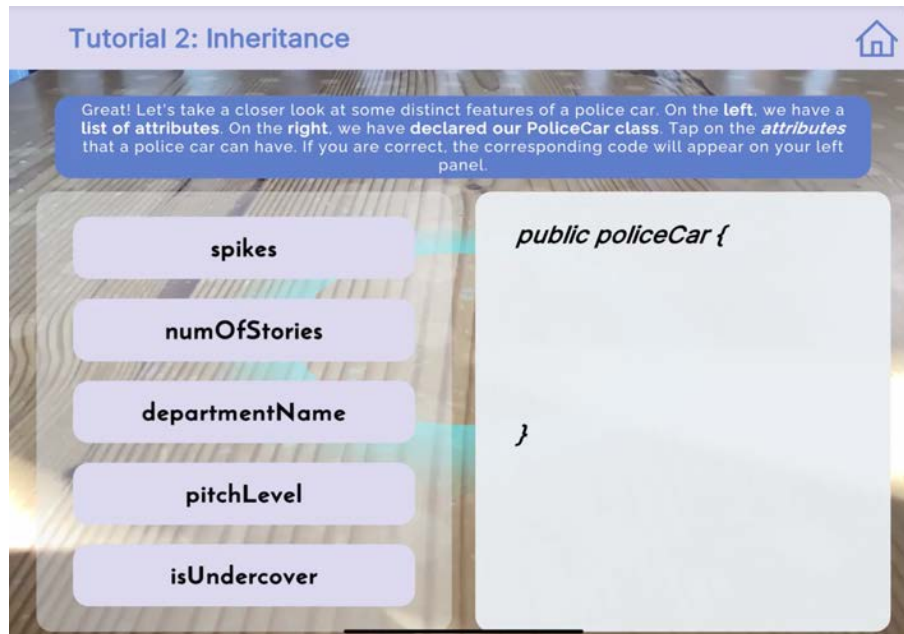


Figure 2.7: Tutorial 2 PoliceCar Attribute Selection

The following exercise deals with the shared methods between Car and PoliceCar. Users are provided with a selection of methods defined in the Car class that they can test on the PoliceCar object in the AR environment. Since PoliceCar is a child of Car, they are able to manipulate it using the same methods defined for Car.



Figure 2.8: Tutorial 2 PoliceCar Instantiation

The final learning outcome for tutorial 2 is to display the greater inheritance hierarchy and briefly introduce the concept of multilevel inheritance. In this exercise, the user gets a chance to see the visual relationship between Car and PoliceCar, and how they fit into the larger inheritance tree with other vehicles such as Truck or SportsCar. By clicking on the parent class box, the user will see the corresponding children appear.

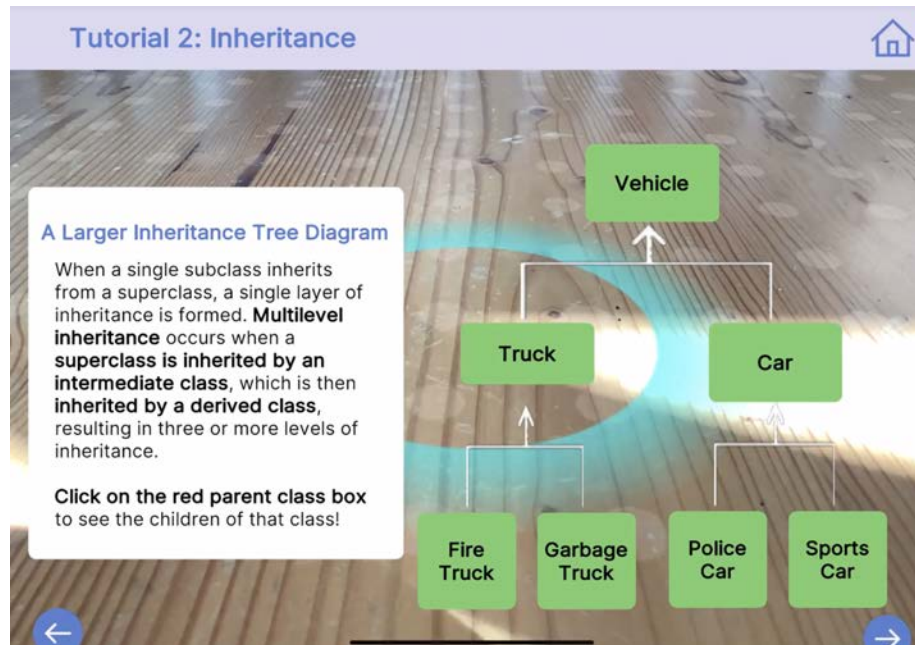


Figure 2.9: Tutorial 2 Inheritance Tree Diagram

Through a series of connected and concise exercises, Tutorial 2 guides students through the broad concept of inheritance.

2.2.3 Tutorial 3: Polymorphism

ORIENT's third and final tutorial focuses on polymorphism. Students will use polymorphism to change the implementation of an inherited method—specifically, they will modify the PoliceCar class's inherited makeSound() method so that it sounds a siren instead of honking.

By the end of Tutorial 3, students should be able to:

1. Understand polymorphism and method overloading, a type of polymorphism.
2. Apply polymorphism by modifying the PoliceCar class so that it has behaviors different from the Car class.
3. Observe the difference between PoliceCar and Car behavior in AR.

Tutorial 3 begins with a series of explanations and examples, which introduce the concepts of polymorphism and method overriding. This accomplishes the first learning outcome and provides students with the knowledge they need to complete the upcoming exercises.

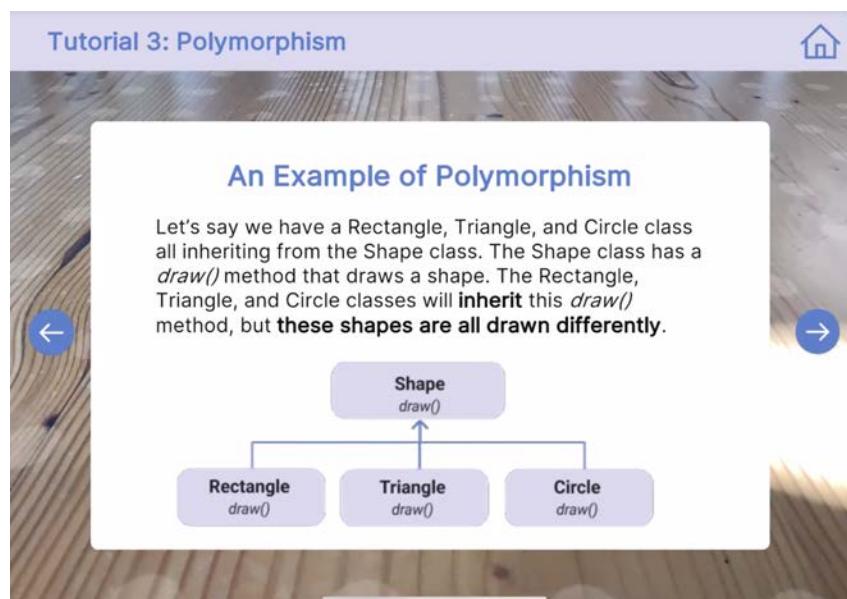


Figure 2.10: Tutorial 3 Polymorphism Explanation

Next, users are introduced to a task that requires them to apply polymorphism to the Car and PoliceCar classes they have created. This fulfills the second learning objective, which is to use polymorphism to modify the PoliceCar class. Specifically, PoliceCar inherits a *makeSound()* method from the Car class, which produces a honking noise. For a police car, the sound used to alert other drivers should be a siren instead of a normal car honk, so we explain that method overriding can be used to complete this modification.

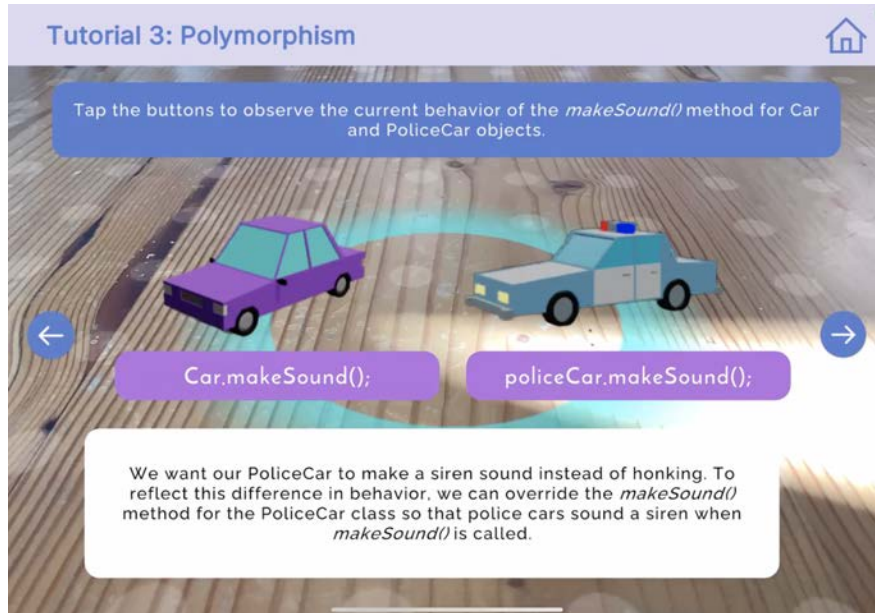


Figure 2.11: Tutorial 3 Applying Polymorphism

Tutorial 3 then guides users to write code that will override the *makeSound()* method for PoliceCars. This exercise involves understanding why modifications are necessary, typing an *@Override* annotation, and filling out the body of the code.

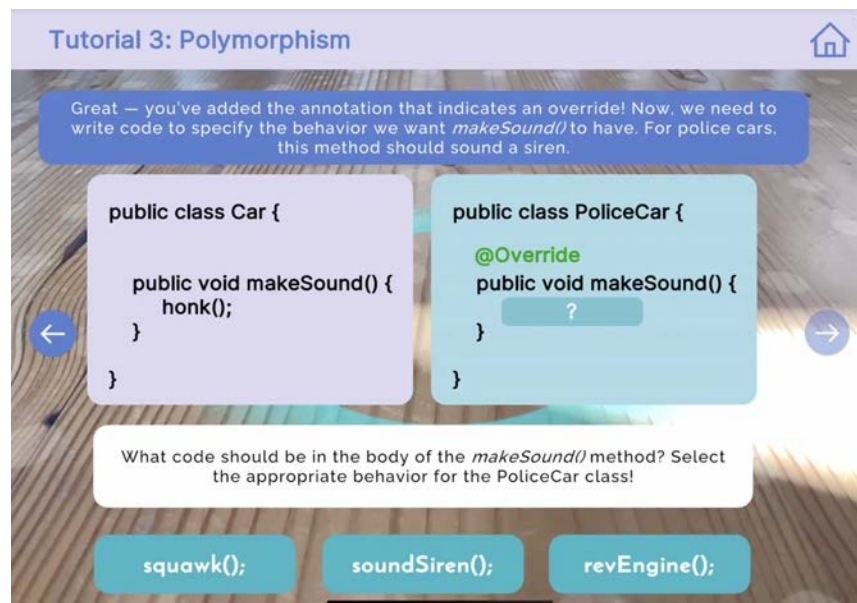


Figure 2.12: Tutorial 3 Method Overriding

The final part of Tutorial 3 lets users witness their code modifications in AR. After typing code to instantiate both a `PoliceCar` and a `Car` object, users can interact with both cars simultaneously. In doing so, they can observe that after overriding the `makeSound()` method for `PoliceCars`, the two cars now make different sounds—one honks and the other one sounds a siren. This exercise fulfills Tutorial 3's third learning objective and concludes the series; as demonstrated, each tutorial builds on top of the previous tutorial, and the end result is that a user can create two distinct types of cars in augmented reality.

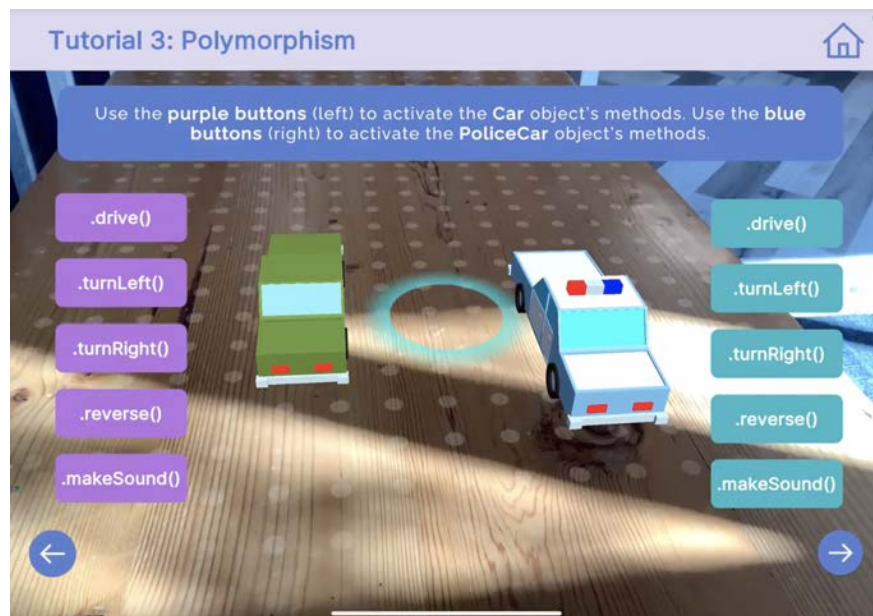


Figure 2.13: Tutorial 3 PoliceCar and Car AR Exploration

Chapter 3

Technologies Used

In terms of technologies, ORIENT was built using Unity, C#, ARKit, and AR Foundation.

3.1 Unity

Unity is a real-time development platform for generating 2D and 3D games and simulations. Unity gives creators the tools they need to create rich, immersive augmented reality experiences that intelligently interact with the real world. ORIENT was created in Unity using the AR packages described below.

3.2 C#

C# scripts are the code files that store behaviors in Unity and are responsible for the engine's overall functionality. Developers can use scripts to generate custom actions and interactions within a game environment. C# is the most compatible with Unity and is the scripting language behind ORIENT's interactive functionality.

3.3 ARKit

ARKit is an augmented reality development platform specifically for iOS devices. ARKit simplifies the process of creating an AR experience by combining device motion tracking, camera scene capture, advanced scene processing, and display conveniences [12]. Since ORIENT is built for the iPad, we used the ARKit development platform to create the AR experience.

3.4 AR Foundation

AR Foundation is an AR framework that enables cross-platform development. Developers can deploy AR apps across numerous mobile and wearable devices. It incorporates both

platform-specific and Unity-specific capabilities. As a result, both Android and iOS devices would be capable of supporting ORIENT in the future.

Chapter 4

Performance Evaluation

4.1 Methodology

To evaluate the effectiveness of ORIENT in teaching object-oriented programming concepts, we performed user testing on six novice programmers. Two students were middle schoolers at the 6th grade level, one participant was a high school student, and the remaining three participants were underclassmen in university. Performing in-depth case studies with select participants allowed us to observe user interactions with ORIENT and collect detailed feedback. Each participant was provided an iPad to explore ORIENT at his or her own pace; after completing all three tutorials, participants were tasked with completing a survey, which asked about their level of education, their familiarity with OOP, and how effective ORIENT was in communicating OOP ideas, in addition to collecting positive and negative feedback regarding the app.

4.2 Empirical Results

In the post-test questionnaire, participants were asked to rate their familiarity with OOP prior to using ORIENT. They were also asked to rate the helpfulness of ORIENT in teaching OOP concepts. The results of the survey can be seen in the figures below.

Prior to testing, three users were very unfamiliar with OOP (familiarity rating of 1/10 or 2/10), and the remaining three users were somewhat familiar (rating of 7/10 or 8/10). After using ORIENT, all testers rated their familiarity as an 8 out of 10 or above, indicating that ORIENT was very effective in teaching OOP concepts.

On a scale of 1 to 10, how familiar were you with Object Orient Programming prior to this tutorial?
6 responses

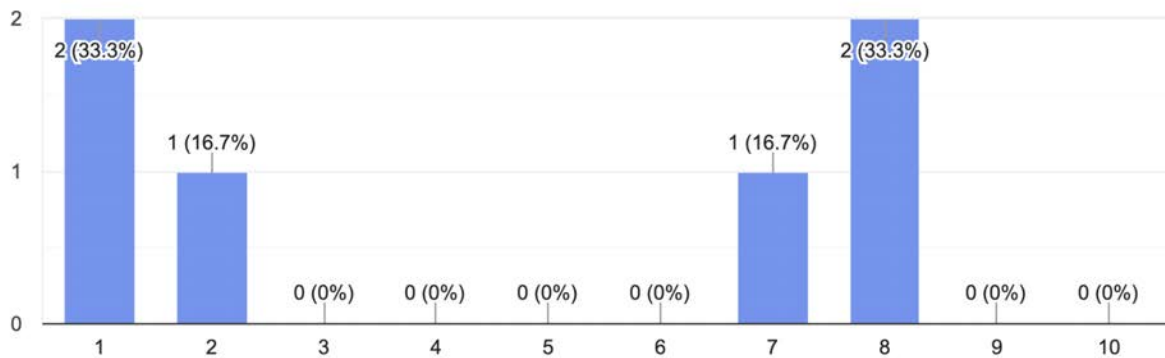


Figure 4.1: Familiarity with OOP before ORIENT

On a scale of 1 to 10, how helpful was this app in teaching OOP concepts?
6 responses

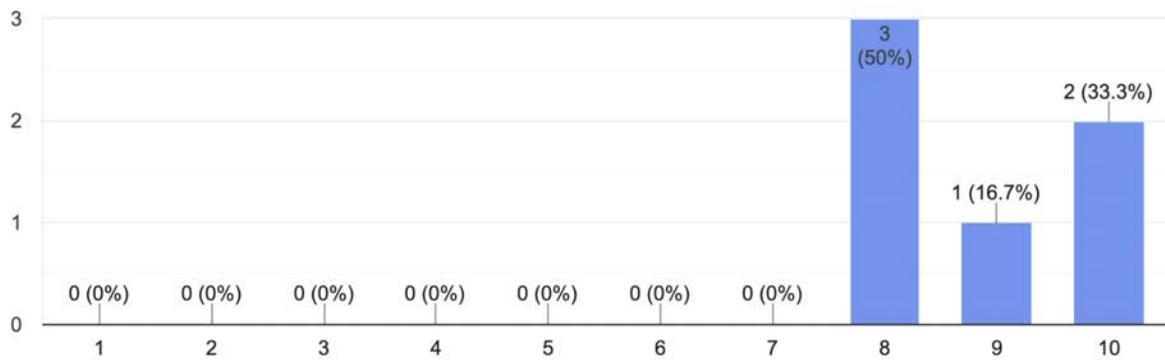


Figure 4.2: Effectiveness of ORIENT in teaching OOP

We also collected data on the length of time it took users to complete the three tutorials. The two middle school students spent 23 and 15 minutes completing ORIENT, while the four high school and college-level testers averaged a time of 10 minutes. Tutorials 1 and 2 each took users about 4 minutes to complete, while Tutorial 3 took an average of 5 minutes and 45 seconds. This data affirms the design of our tutorial series, whereby each tutorial builds upon the previous and increases in level of difficulty.

2.3 User Feedback

To gather feedback on the effectiveness of ORIENT in communicating OOP concepts, participants were asked “What are some things you liked about this app?” and “What are some things you think were difficult to understand or use in the app?”

We then categorized their responses into the following groups based on the initial objectives established prior to project implementation:

1. Present educational content in an age appropriate manner.
2. Build an intuitive AR experience where comprehension of OOP concepts is prioritized over learning vocabulary or code syntax.
3. Encourage students to explore OOP with learning as the primary focus.

The results are summarized as below:

1. Student testers affirmed that they “liked how the application didn’t stress too much code verbatim and focused on user engagement.” Important words were bolded, allowing them to “focus [their] attention to the topics” and crucial concepts. Overall, students found the UI to be “very self explanatory and easy to use.”
2. Additionally, users found the “AR and live demonstrations of the code being written were very helpful in truly grasping an understanding of the material”, and that being able to visualize what they were coding made it easier to see and understand the impact of each line of code. They also noted that the tutorials were “short and sweet” – concise yet also thorough.
3. Lastly, testers “really liked the visuals and how interactive it was,” since this “made it easier to learn and understand the concepts.” Students found “polymorphism with the `makeSound()` function ... great in demonstrating a typically hard concept.” Additionally, a tester said, “I think this app would've been great for introducing me to programming and definitely would've helped me learn the basics much easier.”

In addition to positive feedback, we received feedback on potential areas for improvement. To begin, several students stated that the code-writing activities were challenging for them because

they had to type the code precisely as instructed, including case and space sensitivity. Missing or adding extra spaces caused errors despite typing out the exact words, which was frustrating for students who could not resolve their error.

Furthermore, some of our instructions could have been worded more clearly. For example, in Tutorial 2 one exercise asked the students to “Fill in the blank: public ____ extends ____”. Students were unsure whether to type in the words that fit in the blank or the whole sentence. Thus, by making the instructions more clear we hope to improve the user experience.

Finally, there are minor changes that would make the app experience more pleasant; for example, adding smoother transitions for car movements and removing the AR placement indicator from the background of scenes where it is not needed. Most of these issues could be resolved fairly easily with some additional code revisions.

Chapter 5

Future Work

ORIENT currently includes tutorials focused on class creation, inheritance, and polymorphism, but it does not cover abstraction and encapsulation—two additional major principles in object-oriented programming. Abstraction is used to manage complexity by hiding unnecessary information from the user. This allows the user to build more complicated logic on top of the offered abstraction without having to understand or consider all of the hidden complexity. Encapsulation is the idea of grouping together attributes and methods that work on specific data into a single entity, similar to a Java class. It is often used for information hiding, which refers to the idea of hiding the object’s internal structure and characteristics. Currently, there are no tutorials or exercises available on ORIENT to teach students about the remaining two pillars of OOP. In the future, we hope to expand ORIENT to incorporate tutorials specific to abstraction and encapsulation. This would enable students to complete their introduction to all four tenets of object-oriented programming.

Furthermore, ORIENT can benefit from giving users more creative power. Ideally, users will be able to modify more of an object’s attributes, such as color, number of wheels, size, or the sounds it can generate. This would pique users' attention in the app and allow them to experiment with the endless possibilities of OOP beyond the structured tutorials. This option would most likely be offered after users complete the guided learning tutorials, so as to guarantee that students are still focused on comprehending the various OOP principles of object creation, inheritance, etc.

Finally, by collaborating with local schools and computer science programs, the application can be better aligned with the material covered in a beginning computer science course. Students can benefit from ORIENT’s simplicity and use its visualizations to comprehend difficult concepts. User profiles and login capabilities could also be included in the app, allowing users to track their progress and personalize their learning.

Chapter 6

Societal Considerations

6.1 Ethical Justification

Our rationale for creating ORIENT is supported by two primary ethical frameworks: rights and the common good [13]. As an educational tool, ORIENT promotes the right to education and contributes to the public good by fostering an understanding of how technology in our world works, as well as how it can be created.

Article 26 of the Universal Declaration of Human Rights states that “Everyone has the right to education,” and that “Technical and professional education shall be made generally available” [14]. As such, the availability of computer science education—and educational tools to support it—are critical in our current technology-driven world. As a proof of concept for a mobile application intended to be free on app store platforms, ORIENT makes computer science education more attainable to the public; its use is not limited to classroom settings and can be adopted by any individual interested in learning OOP.

Furthermore, ORIENT’s existence is supported by the Association for Computer Machinery’s Code of Ethics. Section 1.1 states that computing professionals must “Contribute to society and to human well-being, acknowledging that all people are stakeholders in computing” [15]. ORIENT contributes to society by improving the quality and effectiveness of computer science education; learning OOP enables students to pursue computing as a hobby or profession, whereby they may create products that benefit human well-being in numerous ways. Additionally, Section 2.7 notes that a responsibility of computing professionals is to “Foster public awareness and understanding of computing, related technologies, and their consequences” [15]. ORIENT helps novice programmers understand how code works and the effect it can have, thus furthering their understanding of technology’s inner workings.

6.2 Privacy

The expanding use of AR in classroom settings comes with important ethical considerations, especially regarding privacy and data collection. AR applications and devices can track users' locations, movements, and voice, with multiple sensors and cameras; if this information is captured and/or shared, serious breaches of privacy can occur. Although most AR applications in education use mobile devices, the same concerns apply. Cameras can record both the student and the surroundings, so the possibility of surveillance exists. The collection of other data—such as the personal information of young children—also presents a danger. In the United States, the most relevant regulation is the Children's Online Privacy Protection Act (COPPA), which prevents the collection of personal information for children under 13 without parental consent [16]. But while there are laws protecting user privacy, they are by no means comprehensive, and there is a lack of standard policy regarding the development and use of immersive education.

As an AR application, ORIENT does utilize the device camera. However, it does not store any imagery captured by the camera, nor does it support user profiles in its current implementation. If future work were to expand ORIENT, user profiles may be added to support personalized learning and to track progress. In this event, it will be necessary to ensure that user data is handled in a responsible manner and is not shared or used for purposes other than ORIENT's functionality.

6.3 Quality of Education

The use of augmented reality in education is complicated by the unequal availability of technology. Since AR has been shown to be an effective educational tool, those who lack the resources to use it may be at a disadvantage. Schools with more funding can afford fast, reliable Internet and enough smartphone or tablet devices to run AR applications on. Their students have access to innovative AR experiences and tools that may benefit their learning. However, many other students will not have access to such technology [3]. Educational AR applications like

ORIENT have the potential to further the digital divide, making quality of education more closely linked to economic status. However, mobile devices supporting AR functionality are increasingly prevalent; having such a device is the only barrier to accessing ORIENT. With further work, our app would be available on app stores for free and on multiple device types (iOS and Android), removing additional access barriers.

Other ethical issues pertain to the potential harms of using AR in the classroom. One such concern is the “divided brain”: immersive technology may be distracting to students, and the addition of virtual elements may trigger information overload. Moreover, virtual imagery can be misleading, leading to a breakdown of truth. It is unethical for developers to represent ideas or objects inaccurately through AR, because students will not be able to distinguish truth from falsehood. Finally, many digital applications purposefully try to maximize the amount of time users spend on the application, since their business models profit from usage. This can result in digital addiction, which would distract from the learning that AR apps aim to support [16]. To make AR a safe, enjoyable, and effective tool, developers must design and build AR applications in an ethical manner.

ORIENT was created with these ethical considerations in mind; special care was taken to ensure that the application prioritizes learning over gamification or user engagement metrics, and the guided nature of the tutorials serves this priority. There is a clear start and end to each tutorial, and students are intended to use ORIENT simply as a tool to acquire an understanding of OOP. ORIENT is a means to an end, and it is not designed to profit off of increased user engagement. Our consideration of the possible harms of AR in education are taken into account in our project objectives, and can be seen in the metrics we used to assess ORIENT’s effectiveness.

Chapter 7

Conclusion

As the demand for computer science education grows, so does the demand for educational tools to help students better understand and learn material. Augmented reality provides students with a visual and hands-on approach to learning fundamental programming principles. AR can be useful in assisting students in visualizing abstract concepts in object-oriented programming, particularly the principles of class creation, inheritance, and polymorphism.

The proposed solution, ORIENT, is designed to simplify OOP concepts for novice programmers using AR. ORIENT's three main goals are to (1) deliver educational content in an age-appropriate manner, (2) create an intuitive AR experience that prioritizes comprehension of OOP ideas over code syntax and structure, and (3) encourage students to explore OOP with learning as the primary goal. Each of the tutorials reflects these objectives. The application enables students to learn in a novel and dynamic way by providing a guided learning environment, defined learning outcomes, introduction to coding syntax and structure, as well as visible and written feedback at every step of the way. Following in-depth case studies with six students ranging from middle school to university level, ORIENT has demonstrated a strong potential to be utilized in the classroom for introductory computer science courses at the middle school level and above. It is our hope that ORIENT can provide insights on the effectiveness of using immersive technology to enhance computer science education, and we hope that it inspires future developments in this field.

Bibliography

- [1] English, C. (2015). Parents, Students Want Computer Science Education in School. <https://news.gallup.com/poll/184637/parents-students-computer-science-education-school.aspx>
- [2] Arth, C., Grasset, R., Gruber, L., Langlotz, T., Mulloni, A., & Wagner, D. (2015). The History of Mobile Augmented Reality. *arXiv e-prints*. https://www.researchgate.net/publication/275974448_The_History_of_Mobile_Augmented_Reality
- [3] Bitter, G., & Corral, A. (2014). The Pedagogical Potential of Augmented Reality Apps. *International Journal of Engineering Science Invention*, 3, 13-17. https://www.researchgate.net/publication/267153854_The_Pedagogical_Potential_of_Augmented_Reality_Apps
- [4] Scratch. <https://scratch.mit.edu/>
- [5] Kim, S.W. (2017). Development of Additional Functions in Scratch for Learning the Fundamentals of Object-oriented Technology. *International Journal of Applied Engineering Research*, 12(20), 9942-9947. https://www.ripublication.com/ijaer17/ijaerv12n20_98.pdf
- [6] Kodable. <https://www.kodable.com/>
- [7] Screenshot of Alice. [https://en.wikipedia.org/wiki/Alice_\(software\)#/media/File:Alice-2-screenshot.jpg](https://en.wikipedia.org/wiki/Alice_(software)#/media/File:Alice-2-screenshot.jpg)
- [8] Alice 3. <https://www.alice.org/get-alice/alice-3/>
- [9] Alice 3. Scene Editor Overview. <http://www.alice.org/resources/how-tos/scene-editor-overview/>
- [10] Abidin, Z.Z., & Zawawi, M.A.A. (2020). OOP-AR: Learn Object Oriented Programming Using Augmented Reality. *International Journal of Multimedia and Recent Innovation*, 2(1), 60-75. <https://lamintang.org/journal/index.php/ijmari/article/view/83/55>
- [11] Patel, T. (2017). Learning Object Oriented Programming Using Augmented Reality: A Case Study with Elementary School Students. <https://core.ac.uk/download/pdf/97835744.pdf>
- [12] Apple Developer. ARKit. <https://developer.apple.com/documentation/arkit>
- [13] Markkula Center for Applied Ethics. A Framework for Ethical Decision Making. <https://www.scu.edu/ethics/ethics-resources/a-framework-for-ethical-decision-making/>
- [14] United Nations. Universal Declaration of Human Rights. <https://www.un.org/en/about-us/universal-declaration-of-human-rights>

- [15] Association for Computer Machinery. ACM Code of Ethics and Professional Conduct. <https://www.acm.org/code-of-ethics>
- [16] Hawkinson, E., & Klaphake, J. (2020). Legal and Ethical Issues in Immersive Education. 6th International Conference of the Immersive Learning Research Network (iLRN), 305-307. <https://ieeexplore.ieee.org/document/9155135>






ORIENT_Teaching_Object_Oriented_Programming_with_Augmented_Reality_Publication

Final Audit Report

2022-06-08

Created:	2022-06-08
By:	Darcy Yaley (dyaley@scu.edu)
Status:	Signed
Transaction ID:	CBJCHBCAABAabs3NI8iay3QOUdFnyttGEBWWknupIP4g

"ORIENT_Teaching_Object_Oriented_Programming_with_Augmented_Reality_Publication" History

-  Document created by Darcy Yaley (dyaley@scu.edu)
2022-06-08 - 5:51:32 PM GMT
-  Document emailed to N. Ling (nling@scu.edu) for signature
2022-06-08 - 5:52:43 PM GMT
-  Email viewed by N. Ling (nling@scu.edu)
2022-06-08 - 6:34:55 PM GMT
-  Document e-signed by N. Ling (nling@scu.edu)
Signature Date: 2022-06-08 - 6:35:39 PM GMT - Time Source: server
-  Agreement completed.
2022-06-08 - 6:35:39 PM GMT