MYR: A Web-Based Platform for Teaching Coding Using VR

Christopher Berns*
University of Massachusetts Lowell
Lowell, Massachusetts
christopher_berns@student.uml.edu

Grace Chin†
University of Massachusetts Lowell
Lowell, Massachusetts
grace_chin@student.uml.edu

Joel Savitz‡
University of Massachusetts Lowell
Lowell, Massachusetts
joel_savitz@student.uml.edu

Jason Kiesling§
University of Massachusetts Lowell
Lowell, Massachusetts
jason_kiesling@student.uml.edu

Fred Martin¶
University of Massachusetts Lowell
Lowell, Massachusetts
fred_martin@uml.edu

ABSTRACT

MYR is a browser-based, educational platform built to spark student interest in computer science by allowing users to write code that generates three-dimensional, animated scenes in virtual reality. The interface consists of two primary components: (1) an integrated editor, which leverages the MYR API and the A-Frame entity-component-system, and (2) a real-time renderer that displays the corresponding scene. The scenes, which vary in complexity, are viewable using virtual reality headsets, smartphones, and any device that supports a web browser.

By providing access to the specific domain of virtual reality to students, the system aims to make computer science concepts tangible for novice programmers. The MYR development team conducted pilot tests with middle school students in order to collect feedback from this audience. The larger goal of the project is to develop MYR as a research tool to gain insight into computing students' success, motivation, and confidence in learning computing.

The technical implementation, the results of the pilot tests, and the larger vision for future work are discussed in this paper.

CCS CONCEPTS

• **Applied computing → Interactive learning environments**;

KEYWORDS

WebVR, virtual reality, educational tools, research tools, web-based tools, classroom, software

*Undergraduate student research assistant at the Engaging Computing Lab.
†Undergraduate student research assistant at the Engaging Computing Lab.
‡Undergraduate student research assistant at the Engaging Computing Lab.
§Undergraduate student research assistant at the Engaging Computing Lab.
¶Professor and director of the Engaging Computing Group.

SIGCSE'19, February 27–March 2, 2019, Minneapolis, MN, USA

ACM ISBN 978-1-4503-5890-3/19/2…$15.00
https://doi.org/10.1145/3287324.3287482

ACM Reference Format:
Christopher Berns, Grace Chin, Joel Savitz, Jason Kiesling, and Fred Martin. 2019. MYR: A Web-Based Platform for Teaching Coding Using VR. In *SIGCSE '19: 50th ACM Technical Symposium on Computer Science Education, February 27–March 2, 2019, Minneapolis, MN, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3287324.3287482

1 INTRODUCTION

MYR (short for "My Reality") is a beginner-friendly integrated development environment (IDE) for creating web-based virtual reality scenes and experiences. It is built to deliver an engaging experience that can be used by students, educators, and computer artists alike.

We anticipate that providing a means for students to create virtual reality scenes programmatically will foster increased interest in CS for students from a wide range of backgrounds. We believe this can be achieved by providing code libraries, curricular materials developed in collaboration with teachers—such as lessons, activities, and assessments—and technology for CS education in the context of VR programming.

Early on we saw potential in A-Frame[1], a popular JavaScript library for creating virtual reality scenes in the browser, and decided to extend its capabilities with three clear goals in mind:

(1) to create an engaging environment structured around learning text-based programming fundamentals,
(2) to make it available to as many people as possible within a reasonable budget, and
(3) to provide a tool that is capable of aiding research in the field of CS Ed.

Since the MYR editor supports all JavaScript language features, it allows students to engage with common CS concepts, including sequences, loops, functions, events, conditionals, operators, and data. This provides users a platform to create inspiring, creative 3D scenes while simultaneously learning functional, declarative programming.

MYR supports a variety of different virtual reality headsets, including the higher-end HTC Vive and Oculus Rift and the more affordable Google Cardboard. This is important to the educational setting, as it allows institutions to be flexible within their budgets. It also provides an opportunity for interdisciplinary contexts and scenarios in CS Ed that lead to inclusion of many different groups of people in computing.

MYR includes a cloud-based data collection system with which the scenes that have been rendered are captured and stored. This information can then be used to analyze qualitative and engagement-related data, to measure users' learning progress and retention rates, and to research users' problem solving approaches and difficulties. All of this will aid in the development of relevant software features and inform decisions leading to engaging students from many backgrounds.

2 BACKGROUND

Research by Mannila et al. (2006) has shown that programming languages of a lesser syntactic complexity are correlated with fewer syntax and logic errors and thus lead to higher student success rates overall. By choosing a simple language, educators can effectively introduce fundamental programming concepts and skills to potential CS professionals and prepare the foundation that allows for a smooth transition to more complex languages [11].

There are many tools that try to achieve an appropriate level of support for empowering novice programmers. In particular, educational programming languages and environments are engaging to students when they enable them to build their own interactive media—that is, mobile applications, games, experiences, and art [10]. When students use tools such as Logo, Processing, and Scratch to design and implement interactive media, they exercise *computational thinking*, which Cuny, Snyder, and Wing define as "the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent" [7]. The recreational aspect enhances the reward level, opening the door to an exciting world of computer graphics at an early point in a student's CS career [12].

We wanted to build upon this prior work by exploring programming in a reinvigorated medium for creating immersive and highly interactive experiences: web-based virtual reality. Virtual reality enhances the learning experience by providing a world which students can explore—from ideas with which they are familiar to out-of-world ideas which they cannot experience in real life [9]. Previously, utilizing virtual reality technology had certain barriers: accessibility, cost, and technical skill. Now the implementation and adoption of WebVR has made creating and sharing virtual reality experiences highly accessible without set-up requirements, since it can be done through the ubiquitous web browser. Low-cost virtual reality headset technology has made it possible for more people to enjoy the immersive experience virtual reality offers.

There are several existing examples of 3D WebVR digital environments influencing K-12 students' digital literacy development and interdisciplinary lifelong learning [8]. Though these tools are impactful, there is a shortage of tools that balance ease-of-use and sophistication while providing an experience that effectively teaches computational thinking. On one hand, A-Frame by itself provides an inspector environment that makes it easy to create complex three-dimensional scenes, but lacks an intuitive platform to teach CS concepts. On the other hand, an industry-standard game engine like Unity[5] allows a user to program highly sophisticated scenes, but the level of competence necessary to use the program effectively renders it inaccessible to novice programmers. MYR aims to bridge this divide, balancing ease-of-use with education efficacy.

3 OVERVIEW OF MYR

In this section, we describe the design of MYR.

3.1 Entity-component-system

MYR uses the entity-component-system (ECS) of A-Frame [1], a framework developed for creating virtual experiences in the browser using WebVR [6], WebGL [2], and three.js [4]. A-Frame components are reusable and modular chunks of data that users can plug into an entity in order to add appearance, behavior, or functionality. This design pattern has seen recent attention in the field of game development. As described by Unity, "With ECS, we are moving from the object-oriented to a data-oriented design, which means it's easier to reuse code and easier for others to grasp and contribute to it" [5]. An ECS does this by favoring composition over inheritance. This is done by defining every entity simply as a composition of one or more single purpose components. These types of systems are easier to understand as they avoid deep inheritance hierarchies and provide a clear abstraction at both the entity and component level. This is a central reason why the MYR team believes it is likely to be a great experience for the beginner programmer.

In order to make the ECS viable for MYR, it had to be as fault-tolerant as possible. This was accomplished by supplying all of the needed components for a variety of primitive entities with minimal effort. The function call `box()` creates an entity with six different components: a geometry component, a position component, a scale component, a rotation component, a material component, and an ID component, as shown in Table 1.

Table 1: Components in entity created by the `box()` function call.

ID	`box1`
geometry	`{primitive: box}`
material	`{color: 'red'}`
position	`{x: 0, y: 0, z: 0}`
scale	`{x: 1, y: 1, z: 1}`
rotation	`{x: 0, y: 0, z: 0}`

These details are initially abstracted away from the user, but each component can be explored individually. MYR helps make this exploration feel natural and focused on one or two components at a time because they only pertain to a single set of details about the entity. Once users understand how to make a red box, for example, they have a number of different components to explore next. One student may choose to explore the material component, playing with different color combinations and patterns; another may choose to add animations and work on making an animated scene. There are countless ways to explore each entity and expand upon it, another powerful idea behind the decision to build on the ECS of A-Frame.

3.2 Cursor

Fundamentally, a *cursor* is something that preserves state in order to produce subsequent artifacts. In most cases, this leaves some sort of mark on the scene. The process of producing more complicated structures comes from mutations to the cursor rather than the elements themselves.

The idea of a cursor originally manifested in a physical robot turtle which was controlled with a simple API using Logo [12, 13]. The physical turtle became an on-screen turtle in later Logo implementations. This idea of a drawing cursor was later adopted in the Processing language.

When children thought about how to draw with the Logo turtle, they would imagine how their own bodies would move. Papert termed this "body syntonic" thinking, which we can now see as a form of computational thinking [14].

Following the example of these other educational tools, MYR adopts the idea of a cursor for drawing and augments it slightly by using the cursor as a component factory to create the subsequent entities for the ECS.

In order for the cursor to produce a variety of different components, it must have certain properties at all times. By default, MYR supplies all of the parameters to produce all of the components needed by each of the primitive entities it supports. At creation of the entity, the cursor uses its current parameters and creates the appropriate components. The function call `setColor('blue')` will update the MYR cursor with new parameters. The next entity that requires a material component will have blue as its color property.

3.3 An Example

Figure 1 shows a simple MYR program and the resulting image it creates—an ice cream cone. This is done using a global state, or cursor, which will be discussed further.

The following code creates the ice cream cone by:

(1) setting the cursor's color to #A05526 or sienna,
(2) setting the cursor's *X*, *Y* and *Z* position to 0, 1, and -2.5, respectively,
(3) setting the cursor's *X*, *Y* and *Z* scale to 0.5, 1, and 0.5 respectively,
(4) setting the cursor's *X*-axis rotation to 180 degrees,
(5) drawing a cone shape using the current parameters of the cursor,
(6) setting the cursor's color to lightblue,
(7) setting the cursor's *X*, *Y* and *Z* position to 0, 1.6, and -2.5 respectively,
(8) setting the cursor's *X*, *Y* and *Z* scale to 0.5, 0.4, and 0.5 respectively, and
(9) drawing a sphere shape using the current parameters of the cursor.

3.4 The MYR API

The development of the API was a primary focus of the MYR development team. The choice to adopt the notion of a cursor helped significantly. The entire MYR API only ever interacts with this cursor, making it easy to understand. This single point of interaction coupled with the expressiveness of an ECS provides MYR with an API that is both beginner-friendly and expressive.

```
setColor('#A0522D');     // sienna
setPosition(0, 1, -2.5);
setScale(0.5, 1, 0.5);
setRotation(180, 0, 0);
cone();
setColor('lightblue');
setPosition(0, 1.6, -2.5);
setScale(0.5, 0.4, 0.5);
sphere();
```

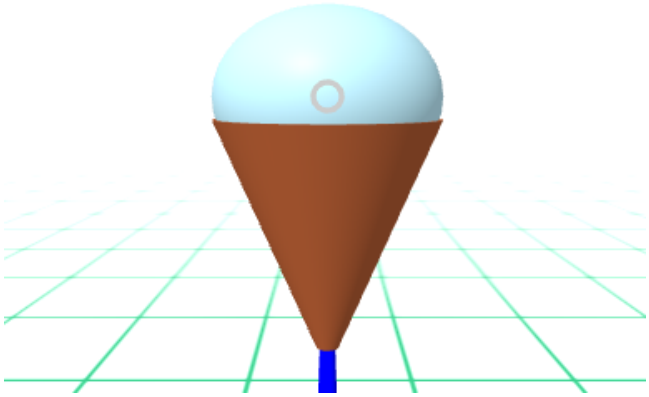

Figure 1: MYR code and resulting image of an ice cream cone

The MYR API also handles some of the more difficult parts of managing the components of each entity. In most cases, components do not interfere with each other, because they are single-purpose, but in certain situations access to the previously defined components is necessary. The expression `fadeIn( box() )` takes the box entity as its first parameter and then adds the fadeIn animation component. The fadeIn animation component handles the shift from an invisible entity to one that can be observed in the scene. In this case, however, the cursor must also access the previously defined material component and add the transparency property to it. This forces the cursor to retain a copy of its previous work in order to access the entities it has previously defined. While this does break the typical notion of a cursor, it is an important change necessary to support the ECS.

Ultimately, it is the control over the interaction with the cursor that helps MYR offer a rich, unified API that is within reach of beginners. It supports many geometric entities and simple animations to be explored, and when coupled with simple constructs, like functions and loops, the user can quickly create building blocks for more advanced entities in subsequent work.

3.5 Groups

In order to make MYR more engaging for advanced users, the team developed *groups*, a powerful feature of the MYR API. Like normal entities, group entities have scale, position, and rotation components. However, in lieu of material and geometry components, they have a component that is comprised of other entities. Groups give the user the ability to create nested entities–black boxes, so to

speak, that can contain other entities or even other groups. Once entities are placed inside of a black box, the user can manipulate the black box as a whole without altering the appearance and relative positioning of the entities contained.

Groups were originally implemented to provide an entity that could be used to simultaneously manipulate multiple entities, e.g. for animations. Subsequently, the team was working to model molecular geometries in MYR, that is, to illustrate the positioning of atoms in molecules. Without the use of groups, the desired models could still be achieved, but altering their scale and position was tedious and and calculation-intensive. By using groups to define the molecules, rescaling and repositioning became easy, because not only did we only have to change the group's scale and position components, but we were also able to use whole numbers for these values.

Convinced that groups could be used in more elaborate ways, we began experimenting with the use of groups inside of functions that returned the group itself. This made the groups dynamic and reusable. By using groups in this way, as well as a simple loop, a crystal structure of molecules can be modeled in MYR. This is demonstrated in Figure 2.

This simple idea of grouping has had a profound effect on the capabilities of MYR. We believe that this will make MYR more enjoyable for anyone who wants to make more elaborate and robust scenes.

```
for (let i = 0; i < 21; i += 7) {
   for (let j = 0; j < 21; j += 7) {
       for (let k = 0; k < 21; k += 7) {
           my_molecule({x: i, y: j, z: k});
       }
   }
}
```

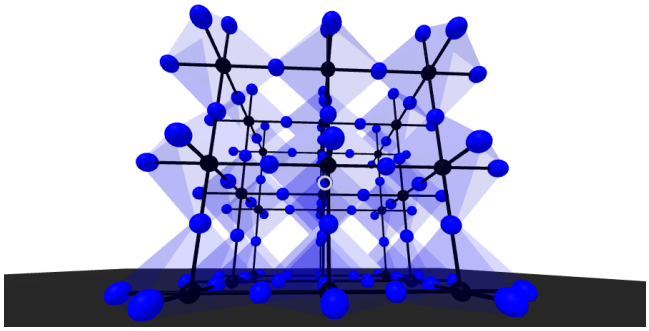

Figure 2: By calling a function that takes a position as an argument and returns a group within a nested for loop, one can model a crystal structure of molecules in MYR.

4 APPLICATION FEATURES

Figure 3 shows the MYR environment in action. The environment runs as a web application inside a modern web browser. The left side is the code-editing area, and the right side is the renderer.

The top row contains the following buttons (from left to right):

- menu to load examples,
- "play" to render a scene,
- "stop" to stop the render of scene,
- "plus" to create a new scene,
- "save" to preserve the scene,
- "folders" to open up previous work and examples,
- "help" to bring up the MYR reference,
- "cog" which offers different view options, and
- login for logging in using a Google account.

In the lower-right corner of the image is the "goggles" button, which converts the display to a full-screen virtual reality render.

4.1 IDE

The MYR application, at its core, is a tool for writing code. Providing a fully integrated development environment is critical to MYR's success. The IDE includes two key components: the editor and the renderer. The editor supports some of the most common features in modern text editors. This includes:

(1) common word processing features, such as keyboard shortcuts, drag-and-drop text, and cut, copy, and paste functionality,
(2) development features such as live syntax checking, code folding, comment toggling, and syntax highlighting, and
(3) a custom-built autocomplete engine to display the MYR API.

MYR's set of features is intended to get users working quickly with tools, and also offer more development-oriented features as well. Exposure helps introduce them to common IDE features and helps them be more productive. Offering a rich feature set helps ensure that the tools are continually able to support the users' best work as they become more proficient.

The IDE also includes a perspective representation of the scene. On a desktop or laptop computer, the scene can be explored with a keyboard and mouse. On a mobile device or virtual reality viewer, the scene can be explored using the device's orientation and screen presses. The user also has the option to leave the editor behind and jump straight into the virtual reality scene itself. Going from the code to the viewer is seamless with just a click of a button.

As with any new tool, there is an initial learning curve. To help ease the learning process, MYR's IDE provides a color-coded reference for its entire API. Learning the API is also facilitated with the autocomplete feature. These features give users an experience that is intuitive and modern.

4.2 Real-time sync

One of the challenges of using virtual reality is the amount of time spent compiling and transferring the scene for viewing. In many cases, this can detract from the enjoyability of the experience.

In practical use, MYR developers edit their code on a desktop or laptop computer, and open the same project on a viewer (e.g. a mobile phone with Google Cardboard). Code changes on the development computer are then immediately viewable on the virtual reality device. The ability to go from code to scene supports an iterative approach, allowing users to frequently test their code.

MYR also provides the ability to publish a scene to multiple clients for viewing. When working in MYR, the user only needs to save the scene to distribute it to all clients. This feature was

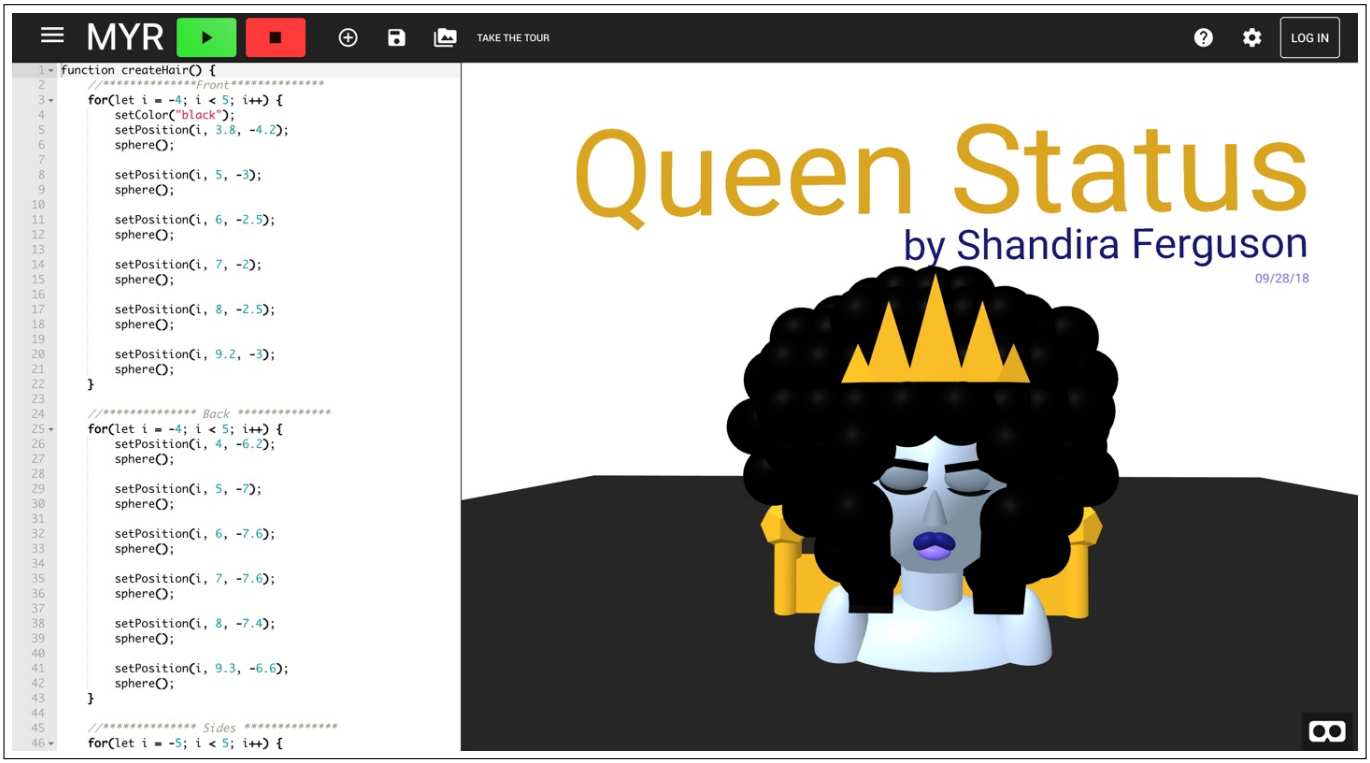

Figure 3: The MYR IDE features the code editor (left) and the resulting scene (right).

developed using Google's Firebase, a serverless architecture service. It provides an easy way for multiple clients to listen for changes in a shared database with minimal effort by the development team.

4.3 Sharing

Learning from example is a staple in many learning environments. The MYR team spent a significant amount of time providing illustrative examples of scenes in MYR. All of these examples are available through a gallery in the IDE. To encourage extension of these examples, the user is able to quickly access, modify, and then save them as their own. This allows the examples to be easily used by an instructor to deliver starter code to students, who can then extend it to meet a certain set of challenges in a curriculum.

The ability to share, however, goes far beyond just delivering samples and starter code. Presently, every project is public and can be shared easily via simple web link. Sharing one's work with others has the potential to keep authors engaged and give them a sense of accomplishment. This approach is intended to build a community of developers and allow them to share their work freely with one another.

4.4 Snapshot capture

While MYR can support imaginative work, it can also be used to conduct research. MYR uses the popular web library Redux for its application state management. By accessing this store, MYR is able to capture the entire state of the application and upload it via HTTPS to its back-end in Firebase.

Snapshots capture the entire state of the application, including the code the user wrote, which user wrote it, and the timestamp at which it was written. The MYR team is then able to produce a timeline from each render to the next. When processing this timeline, certain meta properties arise independent of the code itself.

Perhaps the most significant pieces of data that MYR generates are the rate at which the render system produces errors and the average amount of time between render attempts. These are aspects of particular concern for our current research efforts. The code itself is also useful for gaining insight into different problem solving approaches and even into metrics such as color choice.

5 PILOT TESTING

In July 2018, we ran pilot tests with 13 middle school students as part of a computing camp taught in an urban-rim city in the Northeastern USA. We observed how the students interacted with MYR and gathered feedback about the students' overall experience.

We performed our pilot tests on students who had little, if any, prior experience with either text-based programming or virtual reality. We worked with each student for 20 minutes, presenting them with three scenes and giving them different tasks for them to attempt to complete by modifying the provided starter code. We gave them minimal assistance when it came to task completion.

In the first scene, students were asked complete tasks involving setting colors and using coordinates to place a block between two existing ones. In the second scene, students were presented with

two blocks and challenged to add a third one by using a function. In the third scene, students added a scoop of ice cream to the cone example previously discussed.

5.1 Student performance

Some of the tasks were easily accomplished by students, while others proved to be more challenging.

From the first scene, 11 of 13 students were able to find the correct line number of the code that set the color of a box entity, and 10 of 13 were able to create a third block (8 of them in the correct position).

From the second scene, about function modification, only 5 of 13 succeeded. We realized that functions were a new concept to our test group; none of the students used functions in the third scene task, where such use would have been applicable.

From the third scene, 11 of the 13 were able to add the second "scoop" of ice cream.

5.2 Student feedback

When students were asked what they liked about using a educational virtual reality tool, they stated that the visual aspect improves the overall learning experience. They enjoyed seeing the output of their code, the fascination of creating a new world, the simplicity of using it on a smartphone, the creative control they got over what was displayed in the scene, and simply "everything."

The survey asked students if they were interested in exploring MYR more, ten students responded "yes" and three students responded "maybe." None of our students responded "no." Regarding programming approach, six students preferred text-based programming, five students preferred block-based programming, and two students did not have a preference.

On a scale from one to five, with five being the highest, students were asked to rate how much virtual reality enhanced their experience. Three students rated it a five, seven students rated it a four, and three students rated it a three.

5.3 Takeaways

The results of our preliminary trial were positive. We saw high engagement among all participants. But what was even more surprising was that the enthusiasm persisted, even when they encountered challenging situations. Perhaps our expectations for the students were too high relative to their age group, but despite the difficulty they had completing the assigned tasks, everyone was eager to come back and give it another try.

6 FUTURE WORK

Based on the feedback we received during the pilot testing and from conversations with CS educators, we are developing new features that make using MYR in the classroom easier for early adopters. The following section describes our plans for future work.

6.1 Integrated curricula

One of the biggest challenges the team faced when conducting its pilot tests was delivering the instructions to the user. It was difficult to give consistent guidance from one user to the next. A script could not cover every scenario, and the instructions naturally improved as we learned what works and what does not. The need to provide a consistent interaction from user to user is critical to conducting proper research. Therefore, we plan to develop an integrated curriculum system in MYR, to complement its existing examples library.

As an educational tool, having an integrated curriculum has been demonstrated to be effective (e.g., Codecademy [3] and many other guided programming systems).

The MYR team also recognizes the challenges of creating lesson plans. We are planning a system for the development of different curricula to be integrated into the coding environment itself.

It is further important to draw on the experience of other CS educators. This is why we hope to make this feature available and easy to use, for anyone willing to put together a curriculum using MYR. The MYR team believes that a wide range of curricula is not only beneficial to the user but also to understand how best to teach MYR. With permission of the authors, MYR will provide this curriculum for other instructors in the hope that they will offer a variety of different lessons to suit many tastes.

7 CONCLUSIONS

MYR seeks to deliver the best experience for beginners who want to explore programming and virtual reality. One of our primary goals going forward is to deliver a scalable system that can be deployed in classrooms and also enjoyed at home. We aim to make this tool available to any CS educator, allowing them to supplement their curriculum with an educational adventure into virtual reality.

Inspired by all of the previous work in educational computing, MYR can encourage new programmers by seizing on the recent excitement about virtual reality. MYR was developed specifically to overcome obstacles in virtual reality programming. Features like its real-time sync can be used to alleviate the hardware challenges associated with virtual reality, and by targeting a wide range of headsets, like the Google Cardboard, MYR helps make virtual reality economical for school budgets.

In an effort to investigate the role virtual reality can have in teaching CS, we developed a powerful tool that we hope is adopted by educators. We believe that offering this immersive environment can have a profound effect on overall attitudes towards learning programming. Our initial research looks promising. Students remained engaged and excited about the material. Our results are encouraging as students persisted and maintained positive attitudes despite challenges. Since virtual reality offers an effective platform for immersive, educational experiences, we have the opportunity to bring virtual reality to the classroom.

MYR can be accessed at `learnmyr.org`.

8 ACKNOWLEDGMENTS

We would like to thank Chike Abuah for his significant initial contributions to the MYR project. We would like to thank Vrinda Punj and Elena Izotova for their contributions to the MYR system. We would like to thank Sasha Wilkinson, who reviewed our paper draft. We would like to thank Akira Kamiya for supporting our work with students in the summer camps. This material is based upon work supported by the National Science Foundation under Grant No. 1433592.

REFERENCES

[1] [n. d.]. A-Frame – Make WebVR. https://aframe.io/

[2] [n. d.]. https://get.webgl.org. https://get.webgl.org/

[3] [n. d.]. Learn to Code - for Free | Codecademy. https://www.codecademy.com/

[4] [n. d.]. three.js - Javascript 3D library. https://threejs.org/

[5] [n. d.]. Unity. https://unity3d.com

[6] [n. d.]. WebVR - Bringing Virtual Reality to the Web. https://webvr.info/

[7] Karen Brennan. 2018. New frameworks for studying and assessing the development of computational thinking. (08 2018).

[8] Jorge Ferreira Franco and Roseli de Deus Lopes. 2009. Three-dimensional Digital Enviroments and Computer Graphics Influencing K-12 Individuals' Digital Literacy Development and Interdisciplinary Lifelong Learning. In *ACM SIGGRAPH ASIA 2009 Educators Program (SIGGRAPH ASIA '09)*. ACM, New York, NY, USA, Article 15, 8 pages. https://doi.org/10.1145/1666611.1666626

[9] Soomin Kim, Wookjae Maeng, Cindy Oh, Joonmin Lee, Seo-young Lee, Jeewon Choi, Gil Whan Hwang, Guhyun Hwang, Hyunsung Kim, Joonseok Kim, and Joonhwan Lee. 2017. Immersive VR for Numerical Engagement. In *Proceedings of the 23rd ACM Symposium on Virtual Reality Software and Technology (VRST '17)*. ACM, New York, NY, USA, Article 64, 2 pages. https://doi.org/10.1145/3139131.3141207

[10] Colleen M. Lewis. 2010. How Programming Environment Shapes Perception, Learning and Goals: Logo vs. Scratch. In *Proceedings of the 41st ACM Technical Symposium on Computer Science Education (SIGCSE '10)*. ACM, New York, NY, USA, 346–350. https://doi.org/10.1145/1734263.1734383

[11] Linda Mannila, Mia Peltomäki, and Tapio Salakoski. 2006. What about a simple language? Analyzing the difficulties in learning to program. *Computer Science Education* 16, 3 (2006), 211–227. https://doi.org/10.1080/08993400600912384 arXiv:https://doi.org/10.1080/08993400600912384

[12] Tanya Markow, Eugene Ressler, and Jean Blair. 2006. Catch That Speeding Turtle: Latching Onto Fun Graphics in CS1. *Ada Lett.* XXVI, 3 (Nov. 2006), 29–34. https://doi.org/10.1145/1185875.1185648

[13] Seymour Papert. 1972. On Making a Theorem for a Child. In *Proceedings of the ACM Annual Conference - Volume 1 (ACM '72)*. ACM, New York, NY, USA, 345–349. https://doi.org/10.1145/800193.569942

[14] Seymour Papert. 1980. *Mindstorms: Children, computers, and powerful ideas*. Basic Books, Inc.