Kate Gleason College of Engineering
Rochester Institute of Technology
Rochester, New York 14623

RIT | Kate Gleason College of Engineering

Project Number: P21315

AR/VR Circuit Logic Design

Alejandro Vasquez-Lopez
KGCOE - CE

Ivelin Western
KGCOE - ME

Jessie Chen
KGCOE - CE

Victor Western
KGCOE - CE

Marc Weinstein
KGCOE - CE

Jennifer Indovina
Project Guide

ABSTRACT

Team P213115 was tasked with creating a VR project that was able to teach basic circuit design and construction while also providing the option to perform basic simulations. Due to the nature of the project, it was targeted towards students, academic institutions, and circuit design enthusiasts. This document will be going in-depth on the research, work, and results of the project. To start, the research involved in the creation of this project included reading textbooks and articles that focused on designing something that promoted a positive user experience, looking at examples of VR projects to get a better understanding of how a VR game should be constructed, and looking into what workflow might work best for a project of this scale. Due to the nature of the project, many aspects of the workflow changed as the project progressed, eventually reaching a state that was both efficient and sustainable. The game was developed with the use of the Unity game engine alongside SteamVR. This provided an environment that had a solid foundation to work from.

In the end, the team's efforts have been shared with the client and some of the initial scope of the project was adjusted to meet more realistic goals. This led to an agreement on only having the tutorial scene as our main priority by the end of senior design. This meant having a fully functional breadboard, the ability to perform DC simulations, having basic components implemented, having a detailed and interactive tutorial, and having an environment that felt comfortable to the player. With the documentation provided, the project's progress and decisions made can be easily understood and followed to an extent if someone else was to decide to create their own educational VR project.

Nomenclature

Baked Lighting - Baking lighting calculaulates lighting for static geometry and saves it in textures for later use.

Collider - Collider components define the shape of an object for the purposes of physical collisions. A collider, which is invisible, need not be the exact same shape as the object's mesh and in fact, a rough approximation is often more efficient and indistinguishable in gameplay.

Fitts Law - This law states that the amount of time required for a person to move a pointer to a target area is a function of the distance to the target divided by the size of the target.

GameObject - A GameObject is a fundamental object in Unity that can represent characters, props or other components in a scene. GameObjects do not account for much logic in the game as they primarily act as containers for components, which implement the intended functionality. For example, a light object is created by attaching a light component to a GameObject.

Mesh Renderers - The Mesh Renderer takes the geometry from the Mesh Filter and renders it at the position defined by the object's Transform component. The Mesh Renderer GameObject Component as displayed in the Inspector window.

Light Probes - Light Probes store information about light hitting surfaces in your Scene and provide a way to capture and use that information.

Transform - A transform is a component of a GameObject. It stores several information such as position, rotation, and scale.

Background

The need for new educational tools have been on the rise as the pandemic has shut down classrooms and labs where students gather to learn. For many students in the engineering department, the only way to learn about and test circuits are through labs with physical components. The only existing mainstream alternative to physical circuits are through 2D circuit simulators. These simulators can mimic some aspects of physical creation such as circuit design and simulation, however the experience of placement and component management is lost. An alternative which can help students regain these aspects during quarantine are 3D simulators. VR technology has been on the rise in industry outside of gaming due to the ability to replicate physical experience not found in 2D. The only issue is that there are currently no widely used VR circuit simulators used for educational purposes. To fill this gap a program with the ability to design circuits and simulate circuits with accurate results are required. This need prompted the creation of team P21315. The goal set by the customer was to create a VR circuit simulator which could build circuits, accurately simulate, contain testing tools, provided intuitive tutorial, and gave educational feedback to the player. To achieve this goal multiple decisions were made. The first was the game engine for the project. Unity was chosen due to team experience with programming language and it’s user friendly documentation. The second was the use of SteamVR due to the many existing features and packages it offers. The last big decision was the usage of an open source circuit simulator called CircuitSim which is used in the backend of the system.

Description of Design

Breadboard

Figure 1 shows the first iteration of the breadboard. It's a 25 by 20 pin breadboard with more horizontal space than vertical. To attach functionality to the model we needed points of user interaction. Colliders were placed on each pin in order to record player selection. The collider was programed to toggle on their mesh renderers* when the player hovers over the pins. This was used to create the highlighting feature which allowed for more precision. When the trigger is pressed the mesh renderer is toggled on showing the pin selected. One selection is finished and the mesh renderers are toggled off. One important aspect of selection is accuracy. This aspect prompted various play tests in order to determine the correct scale for the bread or any modifications to the model. The feedback from the playtesters determined that a large scale was necessary and certain aspects of the model had to be changed. One major flaw was the lack of vertical spacing. Figure 2 shows the new breadboard model. This model fixes the issue of vertical spacing and cuts excess horizontal space making the distance between pins more square.

Figure 1: *First Breadboard Iteration*

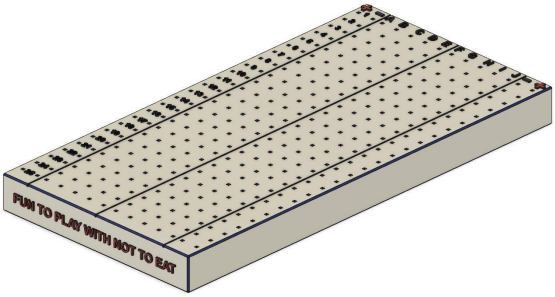

Figure 2: Second Breadboard Iteration

Placement

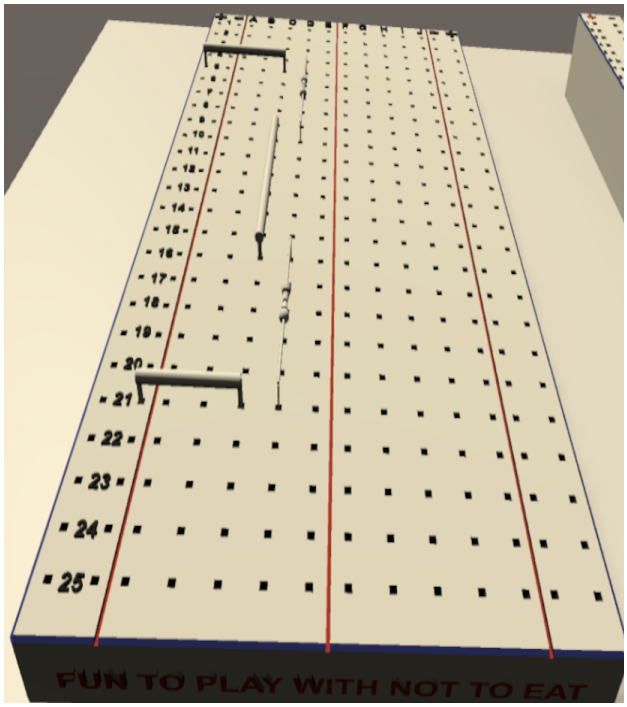

Figure 3: *Component Placement*

The creation of a circuit requires the placement of components. The simulator supports the following components currently resistors, capacitors, inductors, wires, voltage sources, current sources, and LEDs. The functionality of placement was built off of the pin selection feature. Components will be spawn through the UI where you can select the types and magnitudes of components. When spawned they will appear on the player's selected hand This allows the component to follow the hand as it moves. When it's placed, the backend records the pin location. The program then places the component on the breadboard in the correct orientation. The model's legs still need to be attached. To solve this the wire was first implemented. The wire is a 3D cylinder which calculates the distance between 2 pins and scales its x-axis to correct length. This implementation was used on other models such as the resistors to scale the model to the correct length. To finish placement 2 cylinders are spawn at the location of the pins connecting the board to the components

Simulation

As previously mentioned, this project utilized an open sourced circuit simulator called CircuitSim. In order for the team to complete our initial simulation testing, a test environment was created containing UI buttons. This is shown in figure 4. This allowed for basic circuits to be created in Unity. A series of circuits was simulated to test the accuracy of the open source simulator. After the functionality was verified, the team attached circuit logic to our breadboard. This enabled the placement of components as mentioned above which allows users to have an interactive circuit creation experience. Once a complete circuit is created, it can be simulated. During simulation, the results of the circuit are displayed in a popup window that appears on each node of the circuit. This is shown in figure 5.

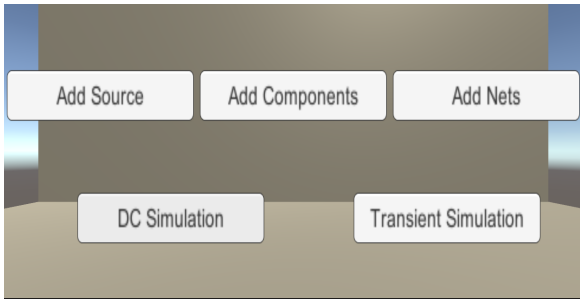

Figure 4: *Simulation Test Environment*

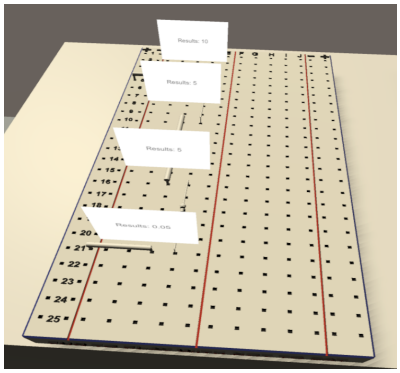

Figure 5: *Simulation Result Popups*

Model

For a substantial amount of time there was no agreed upon design or idea for how the models should look. Some components, such as the bread board and tutorial room, went through several iterations before a final design was settled upon. Past designs tried to do too much, and resulted in models that were functional, but not very appealing to look at for a general audience. That holds true for major components and environmental props alike.

After much deliberation, the team settled on a simple aesthetic that keeps components basic, recognizable, and functional. The bread board, for instance, saw changes to material, color, and spacing between input nodes. This project is in a virtual space, and does not have to adhere to 100 percent accuracy in its visual presentation. The bread board, among other models, is not completely accurate to its real world counterpart. However, its most important and recognizable features are present, making it instantly recognizable for what it is representing. Functionality was not sacrificed in the name of visual presentation. Some visual changes were actually made with the root purpose of enhancing the functionality or improving the user experience.

Other major models, such as the multimeter, also adhere to a simple and recognizable aesthetic. Later components saw significantly less edits and redesigns as the team had settled on an aesthetic, and models generated were designed with it in mind.

User Interface Design

For the user interface, two iterations were used in order to achieve a more robust user experience. In the first iteration, seen in Fig. 7 the layout of the component wall was designed in a manner that would accommodate as many component selection options as possible. This was done to allow the user to have less windows to navigate through to find a desired component. After careful testing and consideration, the team decided that given the space inside the main environment room, it would be ideal to have a smaller window. This in combination with Fitts Law was used to design a new component wall. Thus a second iteration was created. While the user will now have to cycle through more windows, it gives the environment a cleaner design while also freeing up space on the wall. As seen in Fig. 8 the design is much smaller and only focusing on the components that are currently in use. Another detail that was considered when designing the second iteration was the implementation of a multiplier window. This multiplier window would allow the user to select specific components within that range value. Currently this is only implemented for both the resistor and capacitor as there are various levels of resistance and capacitance. Once that multiplier value is set, a final window will appear allowing the user to select their resistors or capacitors within that range. The second iteration also follows a uniform design, meaning that all component windows will be the same dimensions in order to provide consistency for the user.

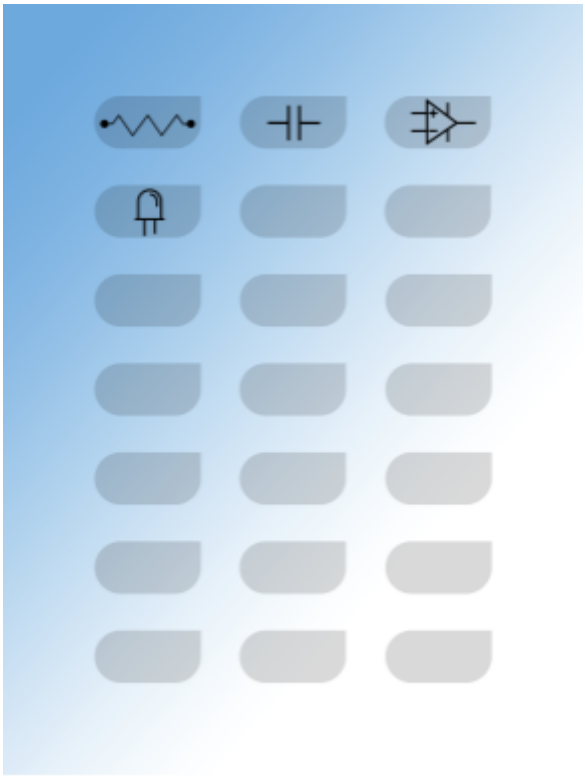

Figure 6: *Component Wall Iteration 1*

Figure 7: *Component Wall Iteration 2*

Currently our component wall supports both laser pointer and controller input for component selection. This is done by aiming the pointer at a specific tile on the component wall and selecting that tile. To take it one step further and reduce arm fatigue, controller input was also completed and this was done via joystick selection. Much like your traditional input system, the user navigates with the joysticks and makes selections with their input button.

Tutorial

For the tutorial, there were two iterations before a definitive style was selected. Figure 10 was the first iteration of the tutorial room and was designed in a way that had the player outside of the room prior to entering it. However, this produced a lot of complications as an area would need to be created outside of this room, preventing the player from falling off the map. This led to the second iteration present in Figure 11. This iteration kept the glass present in the walls, the concept of a doorway, and an open roof. Things of note are that the doorway was planned to have a door in it, and that the beams present in the model were there to provide more excitement in the room. Like the past iteration, there were problems with the rooms concept. One of these issues was the glass on the walls. By having this glass present, the player would be able to look outside, and without a fully developed external environment, it would look odd and out of place. Another issue was with the beams, only adding to the confusion on what theme this room should have.

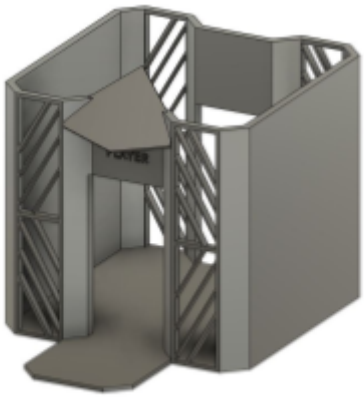

Figure 8: *Tutorial Room Iteration 1*

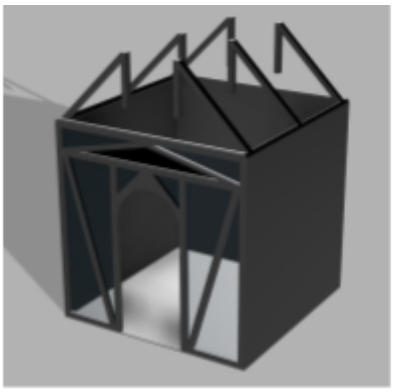

Figure 9: *Tutorial Room Iteration 2*

At this point, it was back to the drawing board, but this time there was an understanding of what did and didn't work. This led to the team deciding on a design that featured a closed top roof, no odd beams, no glass, and logical areas where light could emit from. This concept was then turned into an actual model. This model was then assigned to be the base in which the tutorial room would be developed. At this point in time the team also decided on having a home workshop feeling to it. This proved to be the inspiration for having wood and brick be the primarily textures visible in the scene. From there, props were added to give more life to the scene as a whole, eventually re-introducing the radio from the "Simple Sample" scene. An angle of this can be seen in Figure 12. It should be noted that the component wall was planned to be to the right of the TV present in the scene. After all of that, the room still had some room to improve. Slightly after, lighting was added to the scene, utilizing light probes and the ability to bake lighting into a scene. Reflection probes were also used to make the TV present in the scene reflective, making it more realistic. To add even more of a workshop feeling to the scene, wooden beams were added to separate the roof from the walls. Something important to take note of is that a lot of the aesthetic decisions were made through feedback provided by the client and testing.

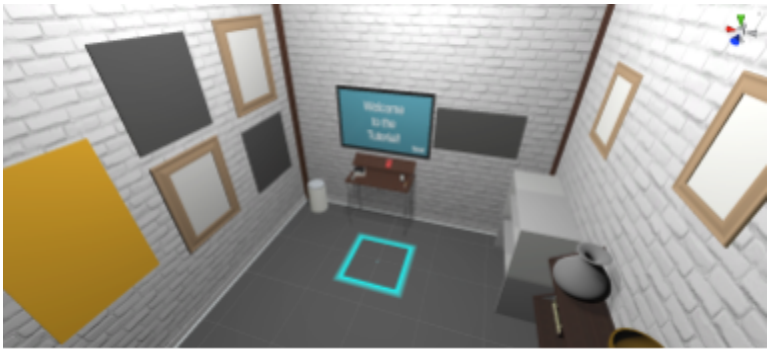

Figure 10: *Tutorial Room*

Table 1: Customer Requirements

Category	Customer Requirement #	Importance	Description
Functional	CR1	1	Software must provide an interactive virtual environment
Functional	CR2	1	Software must be allow users to design circuits
Functional	CR3	1	Software must be able to simulate circuits
Functional	CR4	3	Software must feature multiple components
Feedback	CR5	3	Software must display component details
Feedback	CR6	3	Software must provide feedback to the user
Feedback	CR7	5	Software should provide an in-depth tutorial

Table 2: Engineering Requirements

Importance	Source	Type	Engineering Requirement	Unit
3	CR1	Software	Responsive testing tools	Milliseconds
3	CR6	Software	Player must be able to distinguish what component is selected	N/A
5	CR5	Software	Detail of selected components must appear when selected	Milliseconds
5	CR4	Software	Results and details shown through intuitive UI	N/A
5	CR4	Software	Components must be unique and distinguishable	Number of Components
6	CR7	Software	Player should be guided through circuit creation in tutorial	Milliseconds
6	CR7	Software	Player should be able to obtain information on specific vocabulary word in a fast amount of time	Milliseconds
6	CR1	Software	Player should be able to go through tutorial TV's states in a reasonable amount of time	Seconds
9	CR6	Software	Player will receive visual from interaction in a reasonable amount of time	Milliseconds
9	CR4	Software	Users have access to an adequate amount of components to build simple circuits	Number of Components
9	CR3	Software	Software must be able to simulate the circuit in a reasonable amount of time	Milliseconds

Repository for Version Control

While working with Unity, the team got very used to the provided Unity Collaborate tool. This tool came with limitations and concerns that needed to be addressed. The first of these concerns was the lack of ability to create separate feature branches. The tool had history functionality, but having everything get updated in a single branch led to multiple instances of code and work getting messy and unorganized. This also created merge conflicts within Unity. The next concern was the need for a student plan with Unity. This was a concern as a student plan was required to have access to Unity Pro for free, which is what allowed the team to have a large amount of cloud storage space for Unity Collaborate. This meant that if the project was to be pursued post senior design, a Unity Pro license would need to be purchased in order to keep using Unity Collaborate as the project's size would have been greater than the restrictive 1GB of cloud storage provided with a free licence. This led the team to finding an alternative option for version control, one where storage is not an issue. After some extensive research, the team found something called Gogs. Gogs is a free version control option that allowed the team to create a personal repository that runs on a Raspberry Pi. It utilizes both git and SQL to accomplish this, providing an experience that was similar to using GitHub as a form of version control.

Software

The main software tool used to create this game was Unity as the game engine. Many other software tools were utilized in this project such as ProBuilder which allows for basic modeling and UV mapping from within Unity. This tool in addition to the use of Autodesk Maya and Blender allowed our team to create various models for the game. In order to implement VR integration into our game SteamVR was used in conjunction with Unity to provide VR support for various headsets. Additionally, the backend circuit simulator we used in this project is an open sourced simulator called CircuitSim. In order to create the backend of this game, the team utilized a C# IDE. Initially, the team used Microsoft Visual Studios. However, the team decided to switch to Microsoft VS Code since it allowed us to optimize our workflow. VS Code offered Unity specific extensions, the creation of fewer temporary files, and reduced load times. One of these extensions allowed us to maintain code that is more organized.

References

Valve Corporation. SteamVR Plugin. (2015) [Online]. Available:
https://assetstore.unity.com/packages/tools/integration/steamvr-plugin-32647
Oculus. Oculus Integration. (2017) [Online]. Available:
https://assetstore.unity.com/packages/tools/integration/oculus-integration-82022
Dented Pixel. LeanTween. (2012) [Online]. Available:
https://assetstore.unity.com/packages/tools/animation/leantween-3595
Unity. Probuilder. (2018) [Online]. Available:
https://unity3d.com/unity/features/worldbuilding/probuilder
rami3l. CircuitSim. (2019) [Online]. Available:
https://github.com/rami3l/CircuitSim
Chris Nolet. Quick Outline. (2018) [Online]. Available:
https://assetstore.unity.com/packages/tools/particles-effects/quick-outline-115488
Granby Games. Logic Blox. (2018) [Online]. Available:
https://assetstore.unity.com/packages/3d/props/logic-blox-129409

Acknowledgments

We would like to give thanks to Shawn Foster, our customer, our project guide Jennifer Indovina, our subject matter expert Ethan Harris, and lastly our collaborators Isabel Keirn and Remmington Smith for assisting in the design and modeling for this project.