User Manual

I. Introduction to the Interactive Robotics Education Tool (IRET)

Welcome to the **Interactive Robotics Education Tool**! This web application is designed to help visualize some important robotics concepts through hands-on manipulation. What you will be interacting with was designed by UNC Computer Science students under the guidance of the UNC Computational Robotics Research Group. The goal users (you!) of the IRET will be anyone interested in learning robotics concepts. These users can come from many different backgrounds: students, aspiring roboticists, the curious, and many more! In receiving instruction and specific algorithms from the UNC Robotics team, we know that this application will enhance your understanding of robotic motion and pathfinding techniques.

Before we give you an in-depth look into the IRET, we would like to give a brief introduction to the topics involved with making the simulations. To start, let's talk about **Robotic Pathfinding Algorithms**. A pathfinding algorithm can be defined as a set of instructions that an entity must follow in order to reach a goal. Many pathfinding methods have been developed in order to decrease time needed to find the goal and to select the shortest distance between the entity and the goal. For the IRET, a few specific pathfinding algorithms were implemented for the users. One algorithm is known as the Rapidly-exploring Random Tree, or RRT for short. Later in this manual, we will describe the functionality of the RRT algorithm for you to better visualize what is happening on screen. Another set of algorithms are called the Bug Algorithms. This application showcases Bug0, which will also be further explained later. You can learn about other pathfinding algorithms here, with neo4j.

Another topic is **Robotic Motion Modeling**. As you may know, there are many different types of robots, and therefore many different ways robots can move around. Motion modeling can be described as a type of algorithm designed to visualize the movements of a particular object. In our project, we were given three distinct objects to model: a differential drive (think of a Roomba), a bicycle, and a tricycle. Later in this manual, we will describe these three forms of motion in detail for you to better understand what is happening within the application. You can learn more about motion modeling here, explaining some topics surrounding a bicycle motion model.

In addition to pathfinding algorithms and motion modeling, the Interactive Robotics Education Tool also delves into the crucial concept of **PID Controllers**. PID (Proportional, Integral, Derivative) controllers are a widely-used feedback control mechanism that enables precise adjustment of various processes in robotic systems. Through the IRET's interactive

visualization tool, users will be able to explore and understand the inner workings of a PID controller by adjusting its parameters and observing its effect on a given process value. This hands-on approach will not only solidify users' comprehension of the PID controller's role in robotics but also provide an invaluable opportunity to witness the real-world applications and impact of this essential control mechanism in the ever-evolving field of robotics.

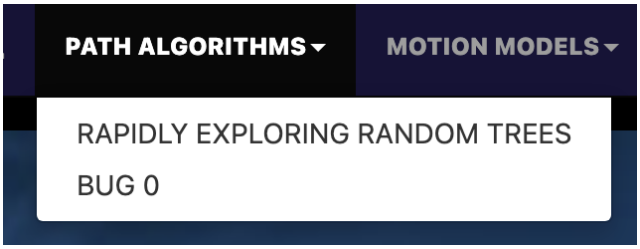
For the next sections of this manual, we will be describing how to navigate our project and further explaining the elements within.

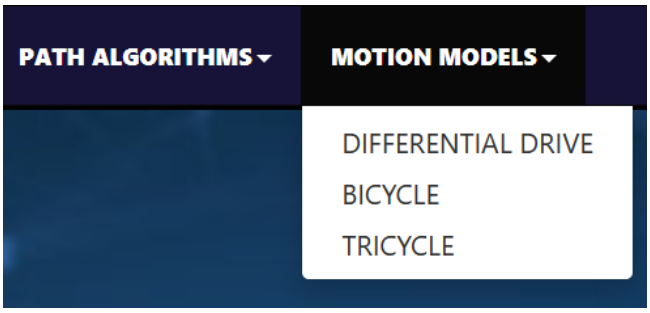
II. IRET Navigation

In opening the Interactive Robotics Education Tool, you will be met with the home screen of the application. This home screen gives a brief overview of what the tool is meant to accomplish. That being a space to experience hands-on manipulation of robotics concepts. The home screen also features our team and a link to our project website for the complete timeline and details surrounding the project.

In order to **navigate** through the project fully, locate the navigation bar on the top left of the page. This navigation bar should look like so:

Within the left section of the nav bar, we can see the title of the application. At any point or section of the application, click on the title to return to the home screen. The right half of the nav bar features two drop down menus to select the visualizations you want to interact with, separated by topic (**PATH ALGORITHMS** and **MOTION MODELS**). When selected, sub-navigation will appear:





From the sub-navigation menus, please select the exact visualization that you would like to interact with and explore.

III. IRET Functionality

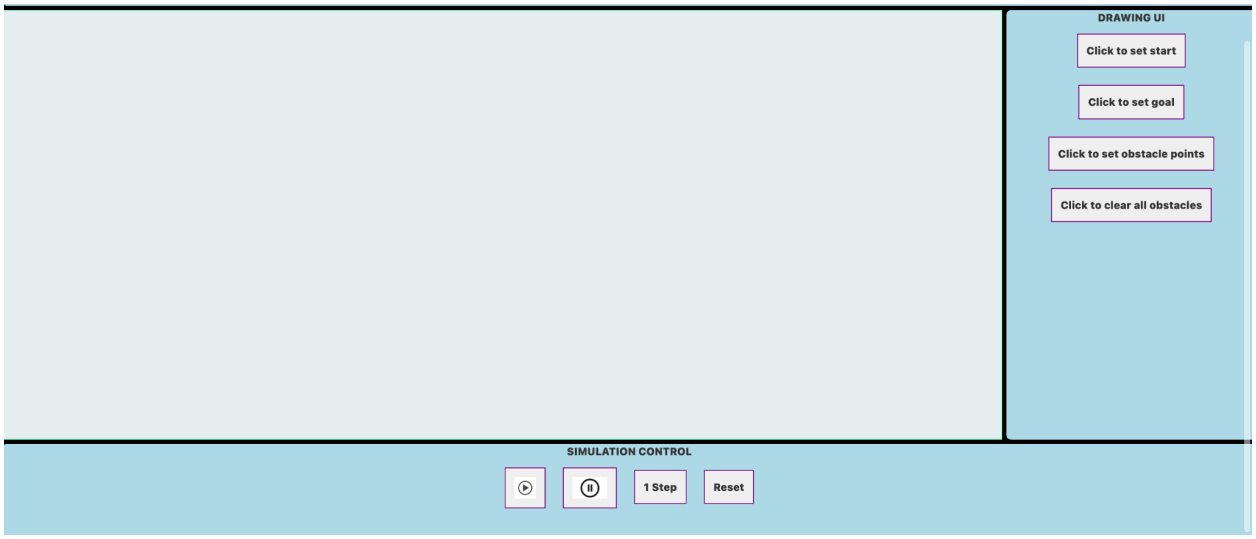
In this section, we will be describing all four of the visualizations by their design and their specific functionality. We have two topics that feature visualizations, Pathfinding Algorithms and Motion Models. Within Pathfinding Algorithms we have one visualization, the Rapidly-exploring Random Tree algorithm. Within Motion Models, we have three visualizations, the Differential Drive, the Bicycle, and the Tricycle.

Pathfinding Algorithms

After selecting Pathfinding Algorithms in the nav bar, select the **Rapidly-exploring Random Tree** algorithm or the **Bug0** algorithm**.** Here you will find a screen with several sections, all working to generate a visualization that you create! There are four distinct sections within the Pathfinding Algorithm template: the **Application Canvas**, **Drawing Control**, **Simulation Control**, and **Informational Header**.

Before we begin, observe the top of the screen and find the section labeled **HOW TO GET STARTED**. This is the **Informational Header** and it is present in all visualizations that we have created. This section is meant to guide the user through initializing and running the visualization, much like this manual. If you ever seem lost or forget a step in the visualization process, return to the Informational Header.

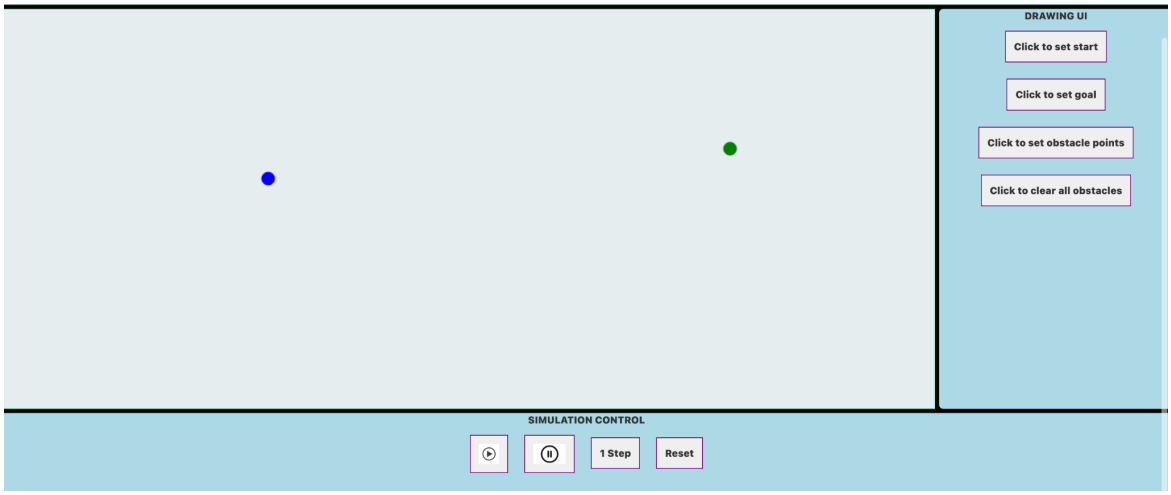
You will notice there are three sections in a pathfinding algorithm: the **Simulation Control** at the bottom of the screen, the **Drawing Control** on the right of the screen, and the **Application Canvas** that takes up the majority of the screen.



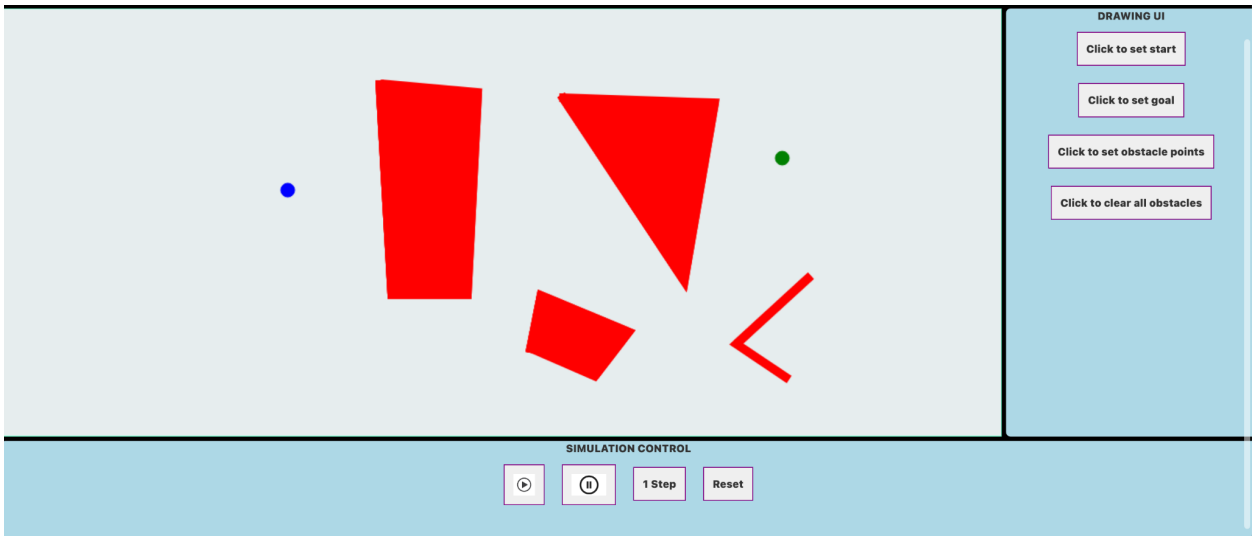
These three sections work in conjunction to create the visualization.

Let's explain the section functionality through a step-by-step process of initializing and running the **RRT** visualization together:

1. Within the **Drawing Control**, find the "Click to set start" and "Click to set goal" buttons. Select the "Click to set start" button and then click some point within the **Application Canvas**. Do the same with the "Click to set goal" button.

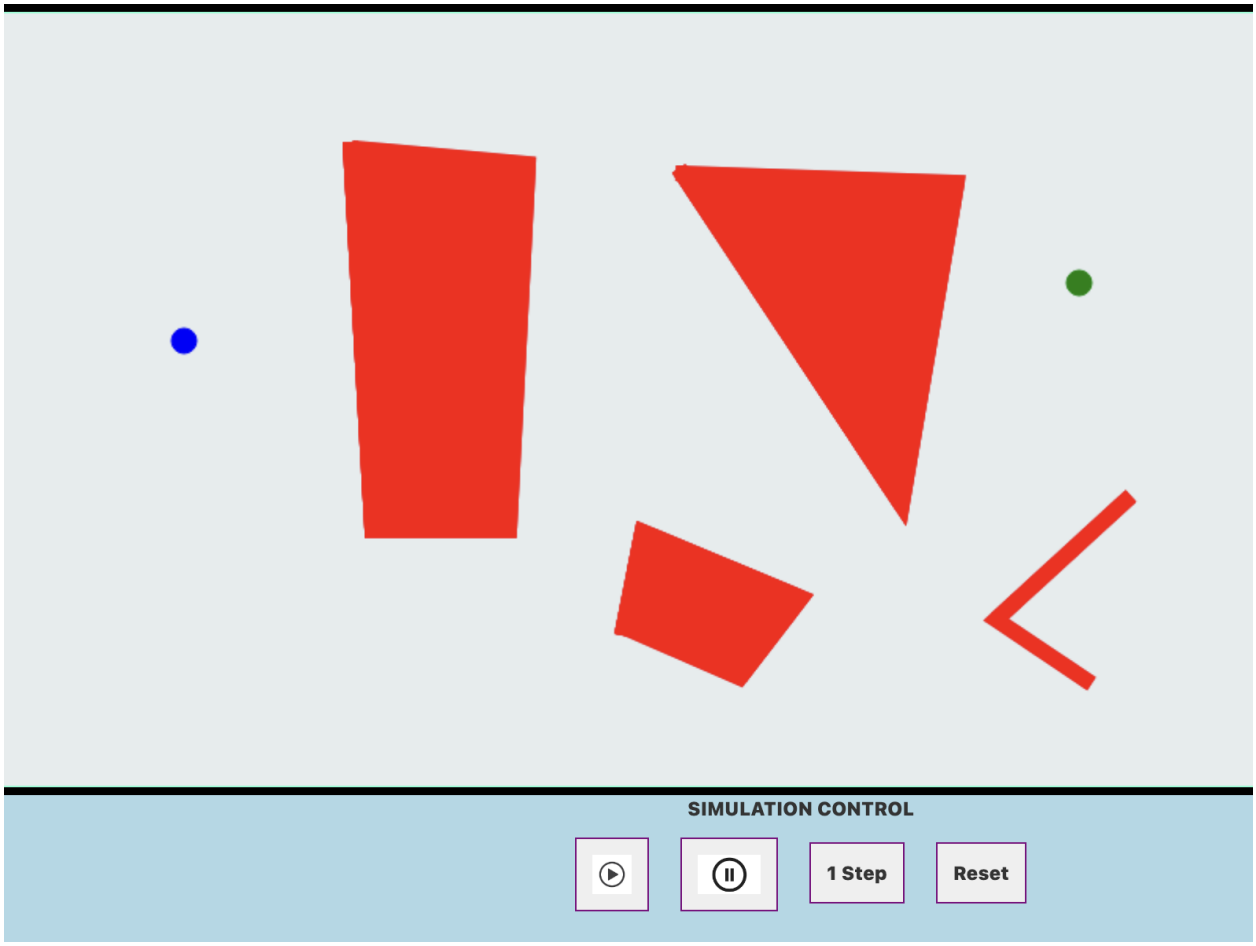


2. Within the **Drawing Control**, click the button labeled "Click to set obstacle points." Then, use your mouse to draw shapes within the **Application Canvas**. These act as obstacles that the pathfinding algorithm has to navigate around to find its goal. As seen, completing a shape fills in its center.

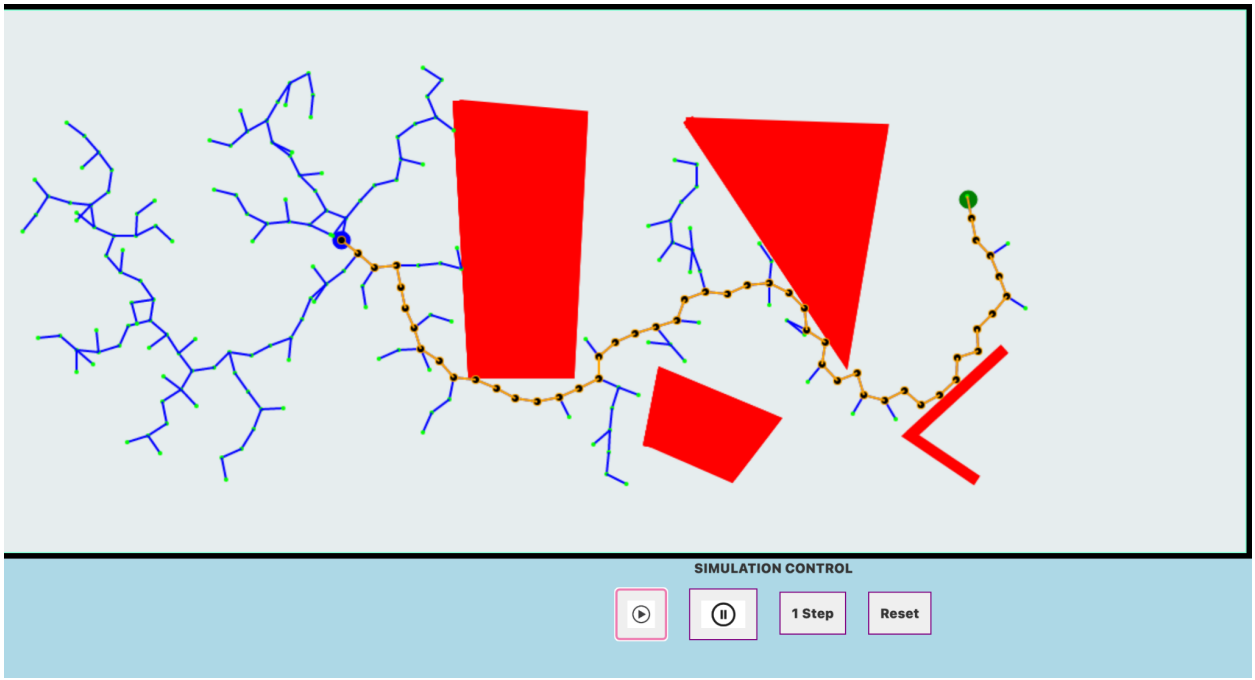


The start indicates the beginning point of the pathfinding algorithm, where the goal is the desired ending point. Using a robot for an example, the start would be the beginning position for the robot and the goal would be the ending position. Here, the start is denoted in blue and the goal is denoted in green. At any time, select the “Clear all obstacles” button to clear both the obstacles and the start and goal points.

Once the desired obstacles and start and goal points are positioned, navigate to the section titled SIMULATION CONTROL. The **Simulation Control** has four buttons in RRT:



The leftmost button is the play button. This begins the execution of the RRT algorithm. The next button to the right is the pause button. This momentarily halts the execution of the RRT algorithm. The button labeled “1 Step” progresses the RRT algorithm by one step. The button labeled “Reset” halts and clears the RRT algorithm completely from the screen. Let’s begin the visualization by selecting the play button:

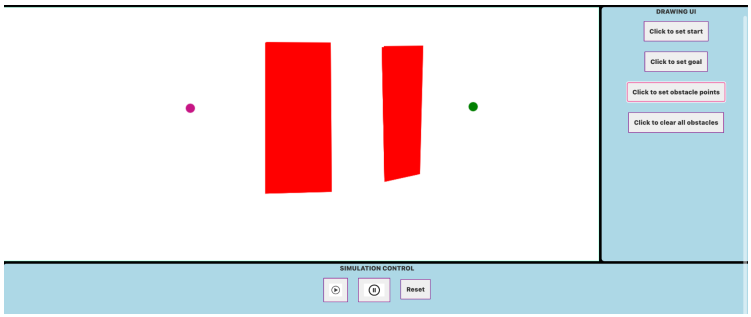


Here, we can see the RRT algorithm at work. The algorithm creates lines of a particular length starting from the start point. This length is known as the “step” length. As one step is made, another step begins from its end, creating a tree-like structure. Each step has its own direction as well, as seen by the tree “branches” pointing in different directions. This branch direction is controlled by a system called goal biasing. This determines what direction the branch will face; if it points toward or away from the goal. When a branch makes contact with the goal, the branches connecting the start and goal points turn a different color to signify that the RRT algorithm has completed its job. At any point after selecting the play button, pause the algorithm by selecting the pause button. Instead of the play button, use the “1 step” button to show each step individually. Use the reset button to clear the algorithm tree.

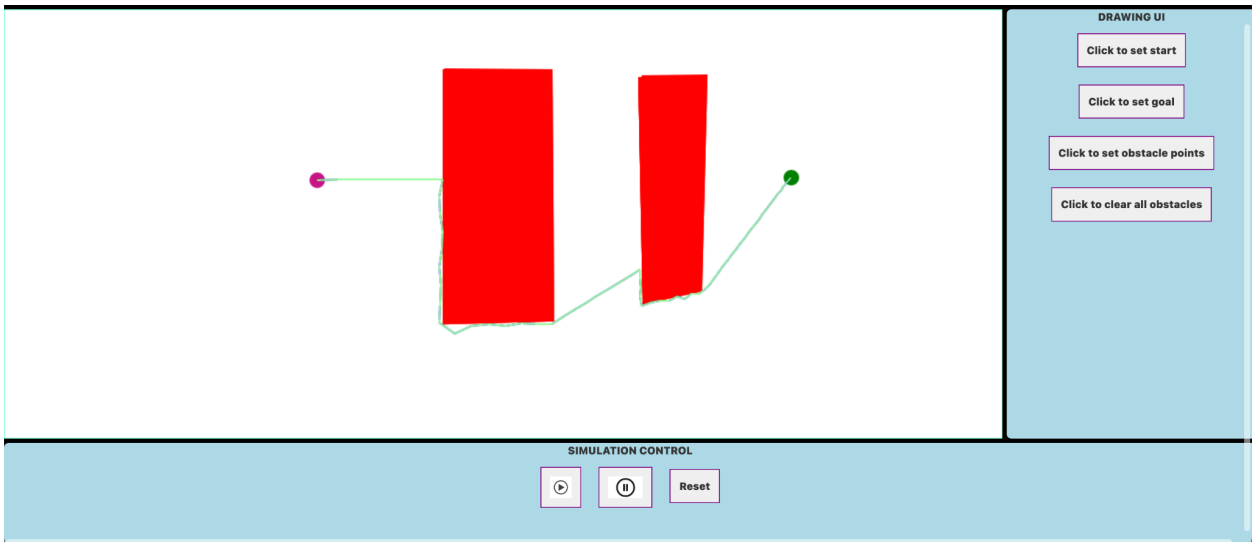
Bug0

Similarly to RRT, you will navigate to Pathfinding Algorithms in the navigation bar, and select the **Bug0** algorithm**.** Here you will find a similar screen as before with several sections.

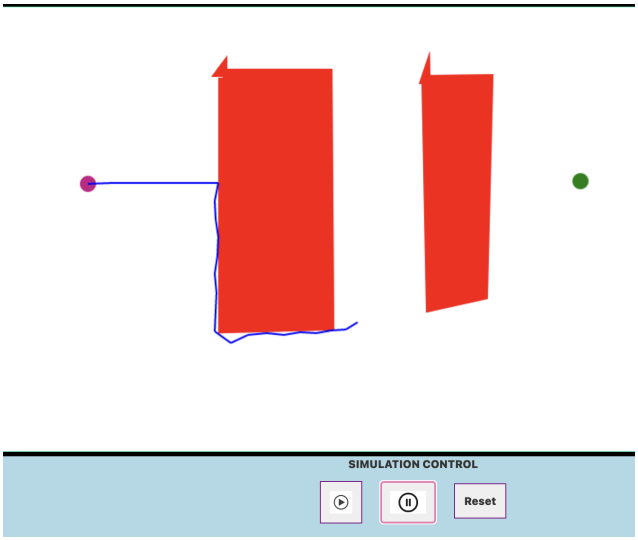
The first two steps in Bug0 are identical to those in RRT. Follow steps 1 and 2 in RRT, then continue reading the directions here.



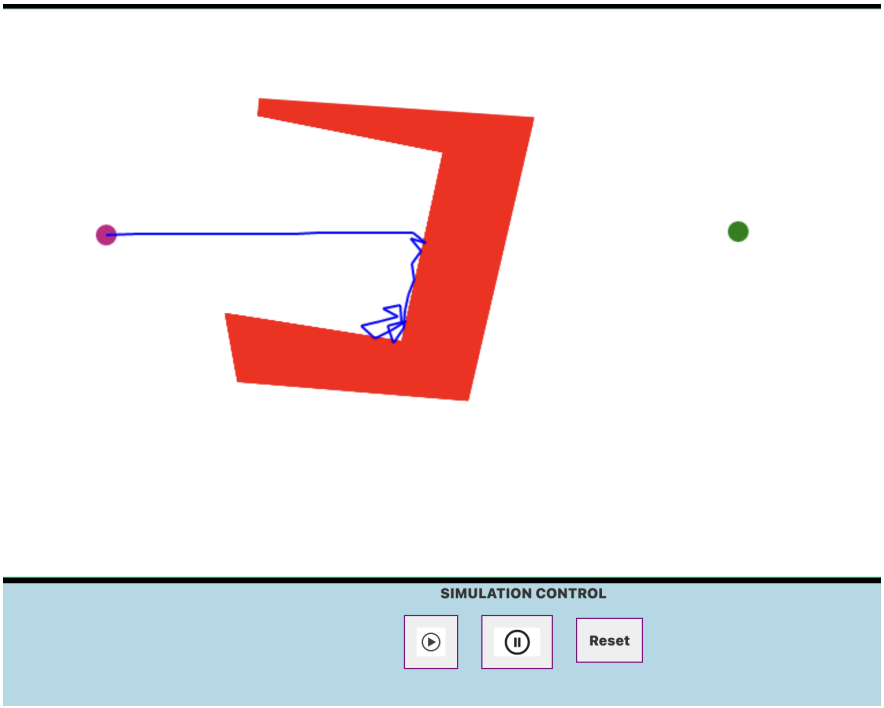
Now, let's navigate to the **Simulation Control** section of the application. Bug0 only has three buttons, as opposed to the 4 in RRT. Clicking the play button will start the algorithm, clicking the pause button will pause the algorithm, and clicking "reset" will stop the algorithm and clear the path. Let's click play, and see what happens.



The algorithm draws a path going towards the goal, and turns to follow an obstacle if it encounters one. Once the algorithm finds a complete path, it will turn pale green, to show it completed. Now, let's pause the algorithm before it completes:



As you can see, the path is a darker blue until it completes. This is useful in cases where Bug0 will break. Let's do an example of a scenario in which Bug0 will not find a complete path:



The path remains dark blue and never finishes, because Bug0 will naively go towards the goal whenever it can. This works in many cases, but there are certain cases (like the obstacle above) that will not allow the algorithm to complete.

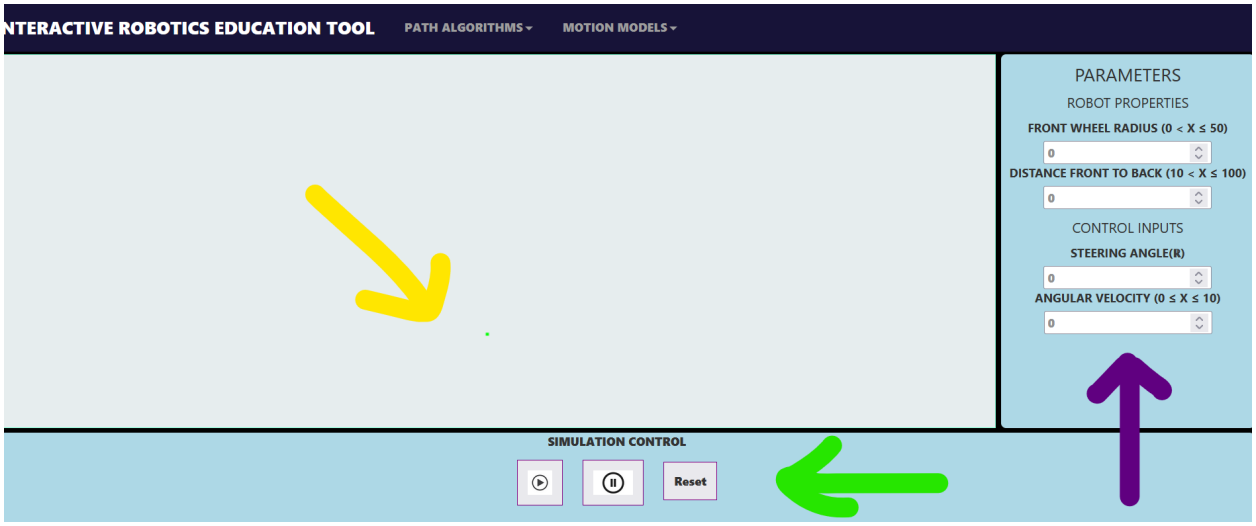
There are many different scenarios to create and explore using combinations of obstacles types and counts, in addition to the placements of the start and goal points. Take this time to explore on your own! What happens when you enclose the start point in an obstacle? How long does it take to find a path when the points are spread farther apart?

Motion Models

After selecting the Motion Models drop down menu, you will see three visualization choices to choose from. These choices being DIFFERENTIAL DRIVE, BICYCLE, and TRICYCLE. After selecting a model of choice, you will be presented with a screen featuring multiple sections. There are four distinct sections within the Motion Models template: the **Application Canvas**, **Parameter Control**, **Simulation Control**, and **Informational Footer**.

Before we begin, navigate to the bottom of the screen and find the section labeled **HOW TO GET STARTED**. This is the **Informational Footer** and it is present in all visualizations that we have created. This section is meant to guide the user through initializing and running the visualization, much like this manual. If you ever seem lost or forget a step in the visualization process, return to the Informational Footer.

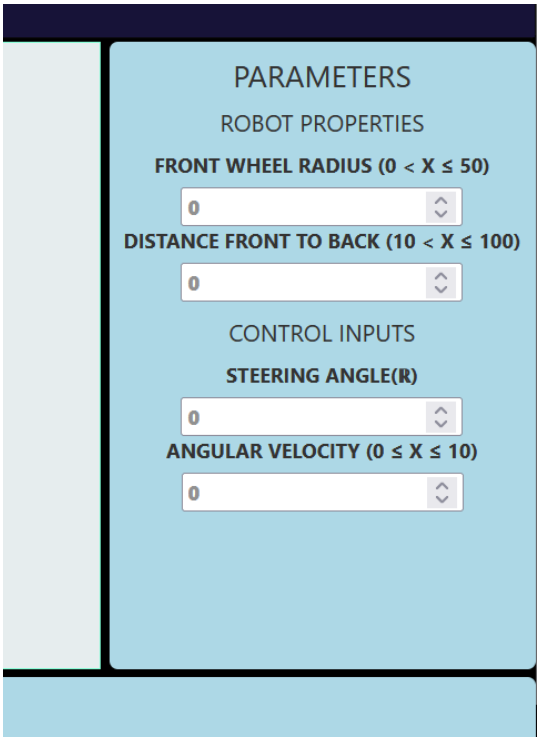
To begin setting up the visualization, navigate to the top of the application window. The **Application Canvas** is denoted by a yellow arrow, the **Parameter Control** is denoted by a purple arrow, and **Simulation Control** is denoted by a green arrow:



These three sections work in conjunction to create the visualization. All sections have the same elements across all three models, except for the **Parameter Control**. Each model has a particular set of parameters, ranging from four to five inputs. For this model, the bicycle, we see that it includes four parameters.

Let's explain the section functionality through a step-by-step process of initializing and running the bicycle motion model visualization together:

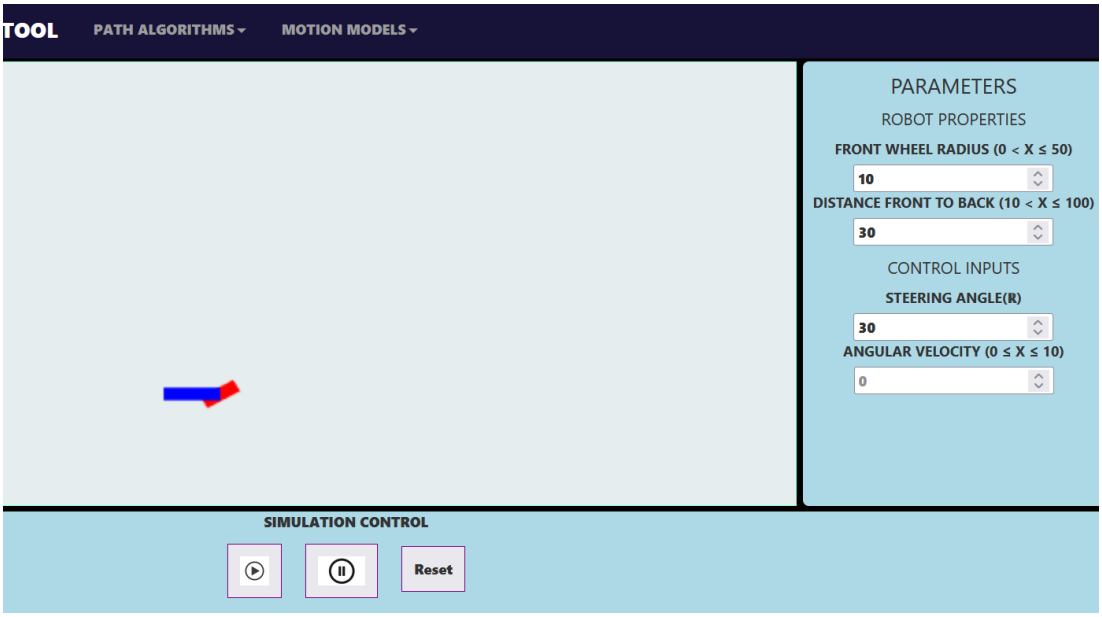
1. To begin, navigate to the Parameter Control on the right side of the screen. Most of the visualization control rests here. We can see within the bicycle model that there are four parameters within two categories:



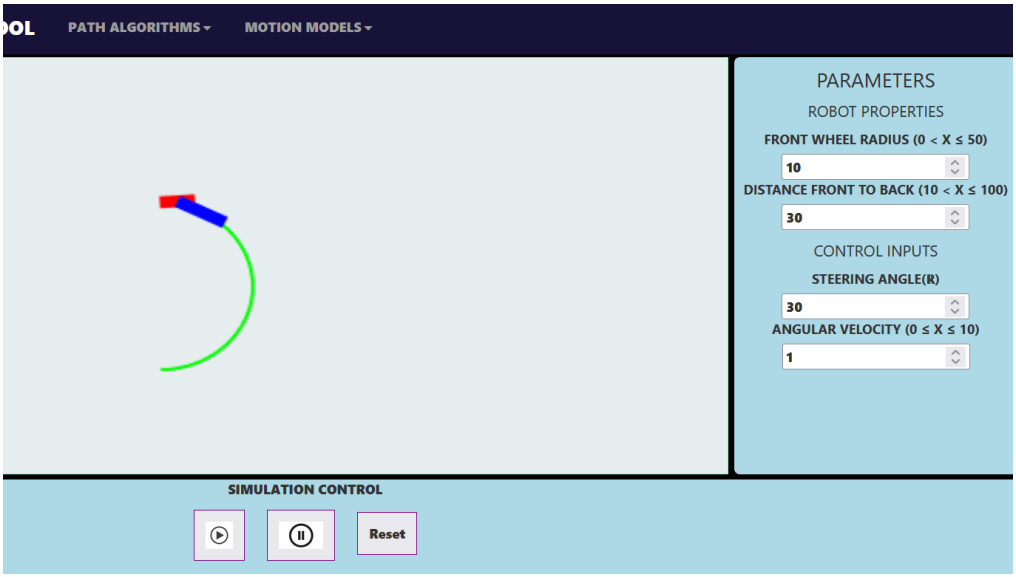
From top to bottom, the parameters are **FRONT WHEEL RADIUS, DISTANCE FRONT TO BACK, STEERING ANGLE,** and **ANGULAR VELOCITY**. Each parameter features a minimum and maximum range of inputs to specify, surrounded by parentheses. The topmost parameter (**FRONT WHEEL RADIUS**) specifies the size of the front bicycle wheel. Under this, the next parameter (**DISTANCE FRONT TO BACK**) controls the length between the front wheel and the back wheel. This changes the length of the bicycle body itself. These two parameters create the Robot Properties. The next parameter (**STEERING ANGLE**) handles the direction that the bicycle travels in. This input takes in a number specifying the angle that the bike turns in. Lastly, the final parameter (**ANGULAR VELOCITY**) controls the speed in which the bike travels. These two parameters control the motion of the robot, making up the Control Inputs.

All of these parameter inputs work in unison to control the movement of the model using a specific algorithm given to us. There are three such algorithms that take in sizing of model elements, speed of the model, and various other inputs in order to present the visualization.

2. For this step, input a number into every input parameter except for the angular velocity. Be sure to adhere to the parameter ranges. This automatically places the bicycle, seen here:



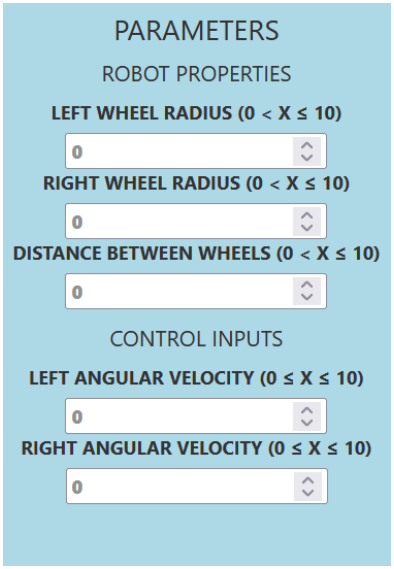
When you are ready to put the bicycle in motion, specify the angular velocity:



Here, we see the bicycle in motion. The travel path is denoted by a green set of dots that track the last position of the bicycle. At any point within the visualization, you can utilize the bottom three buttons within the **Simulation Control**. The middle button is the pause button. Since the visualization begins automatically, you can pause the process by selecting this button. To resume the visualization, select the leftmost button, the play button. To reset all visuals and inputs, select the “reset” button

3. The other two model visualizations, the differential drive and the tricycle, feature the exact same functionality as the bicycle demo, but feature different parameters. Here, we will explain these additions to the **Parameter Control**.

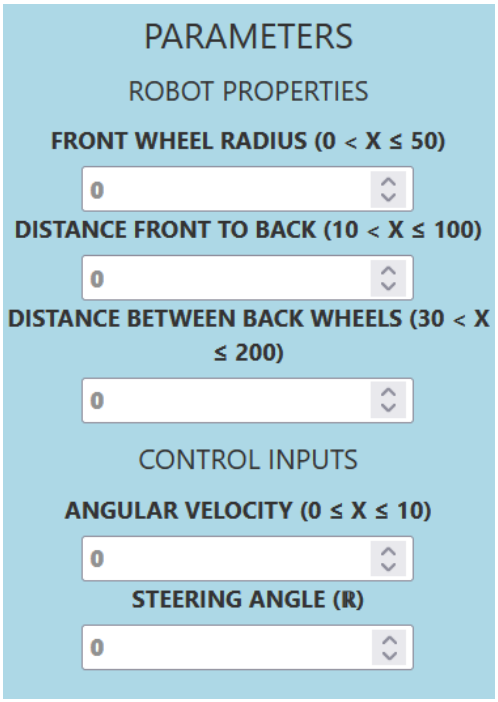
   The differential drive has the following parameters:



   These parameters are **LEFT WHEEL RADIUS, RIGHT WHEEL RADIUS, DISTANCE BETWEEN WHEELS, LEFT ANGULAR VELOCITY,** and **RIGHT ANGULAR**

VELOCITY. The topmost parameter (**LEFT WHEEL RADIUS**) controls the size of the left wheel within the differential drive. The next parameter (**RIGHT WHEEL RADIUS**) does the same, but for the right wheel. Under this, the next parameter (**DISTANCE BETWEEN WHEELS**) specifies the length between the right and left wheels, which determines the length of the differential drive body. These parameters make up the Robot Properties. The first parameter after the Robot Properties (**LEFT ANGULAR VELOCITY**) controls the speed of the left wheel. The next parameter (**RIGHT ANGULAR VELOCITY**) does the same functionality, but for the right wheel. These parameters make up the Control Inputs.

The last model, the tricycle, features the following parameters:



These parameters are **FRONT WHEEL RADIUS, DISTANCE FRONT TO BACK, DISTANCE BETWEEN BACK WHEELS, ANGULAR VELOCITY,** and **STEERING ANGLE**. The topmost parameter (**FRONT WHEEL RADIUS**) specifies the size of the front wheel. The next parameter (**DISTANCE FRONT TO BACK**) specifies the distance between the front wheel and the back axle of the tricycle. This determines the length of the tricycle body. The next parameter (**DISTANCE BETWEEN BACK WHEELS**) specifies the length of the back tricycle axle. These parameters make up the Robot Properties. The next parameter after the properties (**ANGULAR VELOCITY**) determines the speed of the tricycle. The last parameter (**STEERING ANGLE**) determines the angle of the front wheel, specifying the direction of the model. These parameters make up the Control Inputs.

Use these parameters to visualize how each model would function. What happens when the inputs are outside of the specified ranges? What happens when you change distances between wheels?

PID Simulator

Our final topic is the **PID Controller**. In this section, we will introduce the PID (Proportional, Integral, Derivative) controller and its application within the Interactive Robotics Education Tool. The PID controller is a widely used feedback control mechanism that adjusts a process based on the difference between the desired setpoint and the current process value. The IRET features a PID controller visualization tool that simulates and displays the controller's output and its effect on a given process value. Users can interactively adjust the setpoint and PID parameters (Kp, Ki, Kd) in real-time while observing the controller's behavior through an interactive chart.

To use the PID controller visualization tool, follow these steps:

1. Enter the desired setpoint value in the "Setpoint" input field.
2. Adjust the PID parameters (Kp, Ki, Kd) as needed. Start with small values and increase them gradually to achieve the desired controller behavior.
3. Observe the chart as the PID controller attempts to adjust the process value to match the setpoint. Pay attention to overshoot, oscillation, and steady-state error.
4. Continue adjusting the PID parameters to optimize the controller's performance for your specific application.

When tuning the PID parameters, start by adjusting Kp to achieve a balance between response speed and overshoot. Then, fine-tune Ki to minimize steady-state error, and finally, adjust Kd to improve stability. If the system becomes unstable or oscillates, try reducing the Kp, Ki, or Kd values. Experiment with different combinations of PID parameters to find the best performance for your specific application.

For more information on the input and output fields and their meanings, please refer to the PID Controller Testing Guide included in the documentation. It can be found here: src/PID/PID_testing_docs.md.

By understanding and utilizing the PID controller within the Interactive Robotics Education Tool, you will gain valuable insights into how feedback control systems are used to improve the performance and stability of robotic systems.

Important Note: After clicking this tool in the navbar, you will be navigated to a different page as the IRET. Currently, the easiest way to return to the official tool is by pressing the back button on your browser.

***Disclaimer: Because we worked on legacy code, we thought it best to add information regarding our updates in the already existing user manual.