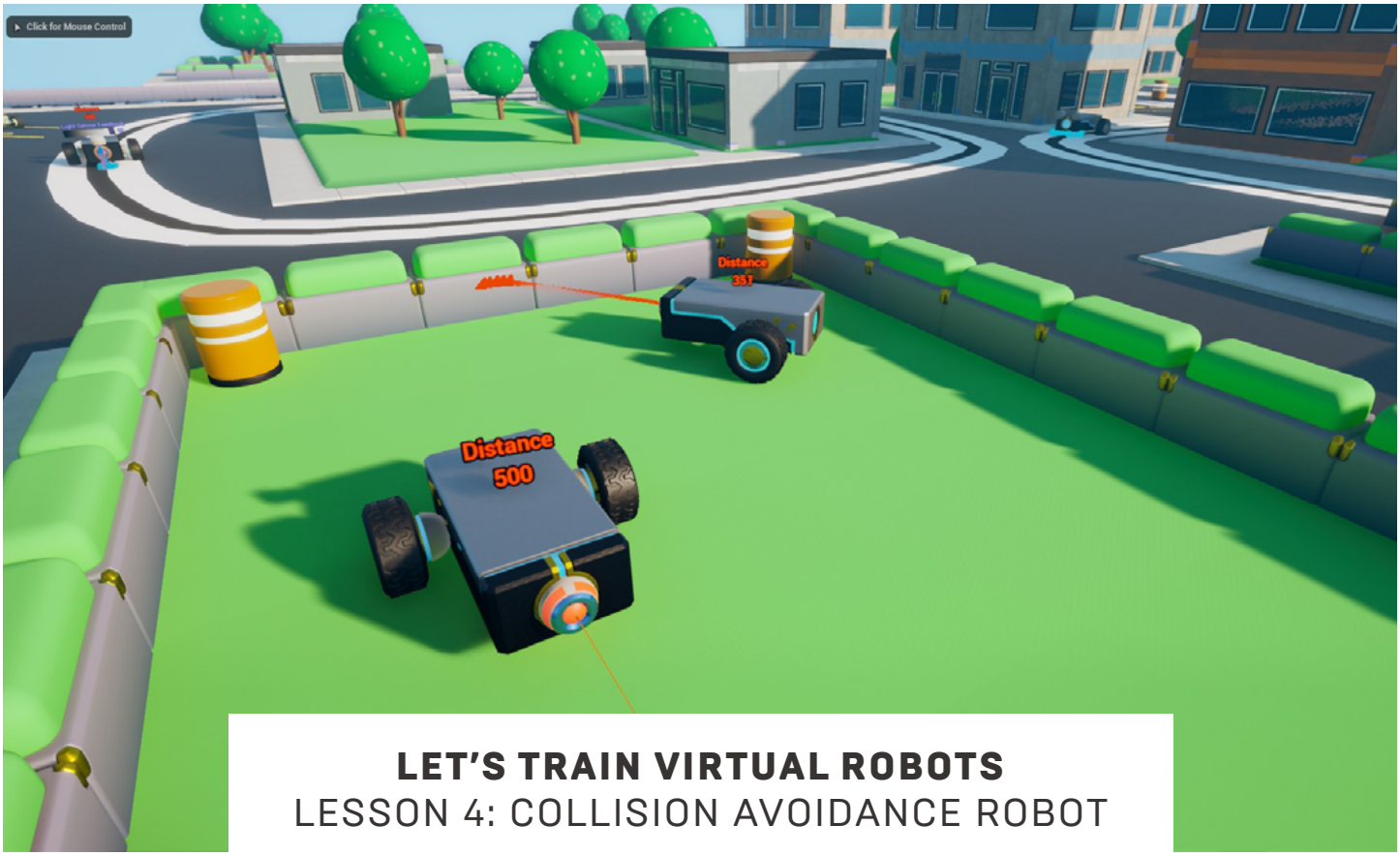
LET'S TRAIN VIRTUAL ROBOTS

LESSON 4: COLLISION AVOIDANCE ROBOT

UNREAL ENGINE

LESSON PLAN

TABLE OF CONTENTS



LET'S TRAIN VIRTUAL ROBOTS
LESSON 4: COLLISION AVOIDANCE ROBOT

LESSON CLASS GUIDE INFORMATION

Lesson Title: Let's Train Virtual Robots: **Collision Avoidance Robot**
Subject: Computer Science, Engineering
Grade: 8–12 (students must be 13 or older to participate in this class)
Lesson Timeframe: One hour
Student Guide

SELECTING ACTIVITIES

These robotics learning activities can be facilitated in any sequence, but it is recommended to proceed sequentially from activity 1 through 5. The lessons work together, building on prior skills, so that a student who completes all five robotics learning activities will learn the basics of coding autonomous robots and how to use the Learning Kit for future activities. It is recommended that all students start with the first activity in order to learn the features of the Learning Kit and how it supports the creation of a robot as well as the coding of motors and sensors in the Unreal Engine environment.

Let's Train Virtual Robots explores the concepts of robotic engineering using applied physics. While giving students immediate feedback on their coding and providing principles/coding language knowledge that they can apply to other physical robotic systems they may work with in the future.

TECHNOLOGY NEEDS

This Unreal Learning Kit requires the installation of the Unreal Engine software and the Unreal Learning Kit files. The minimum hardware requirements are:

- Windows 10 64-bit computer
- Quad-core Intel or AMD, 2.5 GHz or faster processor
- DirectX 11 or DirectX 12 compatible graphics card
- 256 GB SSD (OS Drive)
- 2 TB SSD (Data Drive) <*TO DEVELOP GAMES ON UE*>
- 8 GB RAM

You will also need You will also need an administrator or owner role to install the software.

Installation and setup should be completed prior to beginning the first lesson. For help with installing Unreal Engine, click here.

For help with technical requirements, see this site.

This is a virtual robotic system; no additional robotic parts or kits are required.

AUTHOR CONTACTS

Brian Dickman, Cleverlike Studios

- Author: Brian Dickman
- Email: brian@cleverlike.com
- Twitter: @cleverlike
- LinkedIn: https://www.linkedin.com/in/cleverlike/

Rebecca Parrent, Cleverlike Studios

- Author: Rebecca Parrent
- Email: rebecca@cleverlike.com
- LinkedIn: https://www.linkedin.com/in/rebecca-parrent-5214168/

Ian Southwell, Cleverlike Studios

- Author: Ian Southwell
- Email: Ian@cleverlike.com
- Twitter: @iansouthwell2
- LinkedIn: https://www.linkedin.com/in/southwellian/

Author Brian Dickman studied computer science and operates a full-time game development studio that produces entertaining and educational content inside of popular video games.

Author Rebecca Parent has a Bachelor of Science in Music and Media, along with a Master of Education in Curriculum Design with a STEM focus. Rebecca's work experience includes training adults in computer use and database custom development, and training youth in technology applications and coding.

Rebecca and Brian teach robotics construction and coding as an engaging fun activity that connects to real-world coding languages, coding protocols, and best practices in coding and engineering design. Both authors have shepherded several student teams through robot competitions, specifically in the FIRST LEGO League, FIRST Tech Challenge, and VEX competitions.

Author Ian Southwell attended the Academy of Art University in San Francisco before deciding to take his love for 3D virtual worlds and make it a career. After graduating with a Bachelor in Fine Arts for Computer Animation, he has worked for over a decade as a designer, instructor, and artist. During this time, he worked with seven different studios, gaining Ian an advanced working knowledge of building a game from first concept to final bug testing. He also understands the pipeline of product design, from initial design sketches through a digital sculpt in ZBrush to a 3D-printed object to a cast metal product. He continues to take classes to expand his knowledge and skillsets.

LEARNING KIT WITH UNREAL ENGINE ROBOTICS

Physical robotic systems can be expensive to purchase and difficult to maintain in a busy learning environment. This virtual robotic kit will give students access to a robot that performs with real applied laws of physics, looks like common instructional robot designs and develops skills that can be applied to many robot systems regardless of future hardware and coding languages used.

Robotics is an excellent platform for learning to code because students can see their coding in action. An obedient robot that performs exactly as instructed will provide learners with humorous mishaps and well-earned victories. These activities along with the Unreal Learning Kit will provide a uniquely creative and entertaining coding experience for all students. The simulated physics in Unreal Engine helps students experience the nuances of real-world robotics programming.

Facilitating the Activity

In preparation for facilitating each activity, a video walk-through of the activity is provided by the activities' creators. These walk-through videos are helpful in showing how the lesson can be facilitated and may also be a resource for the students to follow independently.

Through most of the activities, a new concept will be introduced, and students will be able to run a live demonstration of the key learning elements of this concept. In many cases, the demonstration will be followed by a student activity to reinforce the learning by doing the work. The facilitator should review the concept and get student observations/connections/reflections on each concept before moving to the next topic.
The process of introducing a concept, experiencing a live demonstration, and applying the learning to their project will be repeated throughout each robotics learning activity. Upon completion of the

activities, extension activities will be suggested for advanced students or groups that want to extend the activity beyond an hour.

The facilitator should be available for guiding problem-solving as needed by asking questions. What were you trying to accomplish? What steps did you take? What did you observe? What can you try next? Review the Engineering Process for inspiration.

Students can work independently, or in small groups as needed. Consider recruiting the assistance of students that have shown proficiency with the activities to help aid their peers.

ACTIVITY FOUR – COLLISION AVOIDANCE ROBOT

Introduction

Safety first...avoid a collision!

Smart cars and drones (whether underwater or airborne)use sensors to detect objects that can be perceived as a target destination, or as an obstacle to be avoided. How do these sensors detect objects nearby? One powerful sensor known as LiDAR (pronounced Lie-Dar) uses a laser to measure distance. It's actually similar to our light sensor because it uses a laser to emit a beam of light over a long-distance and a sensor that detects reflected light. Instead of measuring the light level, it computes the distance based on how long it took the light to travel from the emitter and back to the sensor.

Even some iPhones use a LiDAR sensor3 to enhance video focus and provide measurement data to apps.

LiDAR (which stands for Light Detection and Ranging) is used in many real-world applications. It is capable of accurately measuring distance with a narrow pinpoint focus and can be programmed to scan an area and use the data to detect the size and motion of an object. This sensor is so powerful, it can create a detailed map of terrain from an airplane. [1], [2], [3]

Activity Overview

In this activity, students will learn how to use a distance sensor to detect objects in the path of our robot. The robot will be programmed to follow a line, just like in *Let's Train Virtual Robots* lesson three. But this activity will upgrade our robot to stop if an object gets in its path. Capabilities like this could save lives while advancing technology!

All the functions of a LiDAR sensor are beyond the scope of our activities, so we have created a simplified version that measures distance. The distance sensor in the activity acts like a laser distance measuring device and will return the distance of the first object in its direct line of sight.

For the collision avoidance task, when the distance sensor detects an object in the path of the robot it will stop until the object is cleared. When the robot no longer sees an object in its path, it will resume

moving forward.

Deliver the pizza or package, but don't run into anything or drop the delivery!

Coding Connection

For the robot to receive the information, the flow of code must contain a loop and a conditional statement, where the sensor continues to test the result and make adjustments to avoid collisions and continue moving forward. With two different sensors on this robot, there will be two conditional statements. These are commonly called "nested" statements because the second conditional will only run if the first one is true. The robot will continually run until the student stops playing the program.

Essential Questions

1. What sensor will be used to detect objects in front of the robot?
2. What direction should the sensor be facing?
3. How will the robot know there is an object in the way?
4. What actions will be taken when an object is blocking the way? What happens when the object is gone?
5. What sensor is used to follow the line on the ground?
6. What conditional statements will be used by the robot to follow the line and avoid collisions? Explain verbally to demonstrate understanding of the logic. *(<type of sensor> will detect <condition> and execute <action> otherwise it will <action>)*
7. What are the shortcomings of using a single laser distance sensor to detect objects in front of our robot?

Learning Outcomes

After completing this lesson, the student will be able to:

1. Code a robot to "see" an object with a laser distance sensor.
2. Code a robot to drive forward continuously while looking for an object in front of it.
3. Code how a distance sensor provides feedback, and how that information is interpreted by the robot.
4. Use the Blueprint programming language to implement a loop and conditional statements that make the robot avoid collisions while constantly following a line on the ground/surface.

Getting Started in Unreal Engine

In order to be successful with this Unreal Learning Kit activity, participants will need to learn the basics of navigating the Unreal Engine interface. Improved proficiency in navigating the player/camera and locating the panels/windows inside the application will create a richer learning experience. Each activity will review the essential interface components and controls needed for completing the exercise.

If your participants are not familiar with Unreal Engine, consider allocating more time for the initial activity or plan some preparation exercises prior to starting the activity.

Scope of Unreal Engine Usage

- Launching Unreal Engine and opening the sample project
- Identifying and activating various levels/maps
- Playing your code and exiting Play mode
- Navigating your camera/view in the Viewport
- Creating your own robot by duplicating the original
- Moving and rotating actors in a level
- Editing default robot settings
- Saving your project

ACTIVITIES

Technology is constantly upgrading to the next best thing. It's time to upgrade our self-driving robot! Instead of solely focusing on the line, we will add the ability for the robot to avoid colliding with objects in its path. To do this, we will need to add a new sensor and modify the code to use multiple conditional statements.

Line-Following Robot

If you created a line-following robot in activity three, we can upgrade your robot. If you would like to start fresh, a working robot is available in the sample project. This activity will start with a working line-following robot that cannot avoid collisions.

Activity 1: Add a self-driving robot to your level and make sure it can successfully follow the line.

Adding a Distance Sensor

Let's Train Virtual Robots includes a distance sensor that returns the distance of the nearest object in the direct line of sight from the sensor. The sensor has a maximum effective range, so it can only detect objects within a specific distance.

Activity 2: Review the example level that shows the various functions of the distance sensor.

Detecting a Possible Collision

As we get feedback from the sensor, we need to determine what distance should be considered "too close." If an object is anywhere within this distance, the robot should stop.

Activity 3: Stop a robot from crashing into a wall. Equip the robot with a distance sensor and create a conditional statement that will stop driving before hitting the wall. Note the speed of the robot motors and the distance required to safely stop without hitting the wall.

Combining the Driving and Collision Avoidance

We should now have the functionality to follow a line AND avoid a collision. Next, these functions must be combined into a single robot that will follow a line and stop if an object is blocking the way. This challenge will require nested conditional statements in the code. It will be easier to implement if you have an understanding of the solution prior to coding. (This is true for most coding.)

Challenge: Add a distance sensor to a line-following robot and use nested conditional statements to successfully follow the line, stopping for any obstacles that get in the way.

Extension Activities

To reinforce the skills developed in this activity, students can extend their learning with multiple additional challenges.

- Add additional distance sensor(s) to improve collision avoidance
- Add additional light sensor(s) to improve the ability to follow the line
- Try to increase the movement speed of the robot. What happens? What changes need to be made?
- Try creating a longer and more complex line for the robot to follow

Activity Review

- Learn to use a distance sensor
- Use feedback from the distance sensor to avoid collisions
- Determine how to use sensor feedback to decide when to stop the motor
- Combine line following and collision avoidance into a single robot using nested conditional statements
- Solve more complex logic by making a plan

External Resources

- Observation log – blank one, and one for the Extension Activity
- Navigation Cheat Sheet
- Threshold Calculation Worksheet

ASSESSMENT

Rubric

Concept	Distinguished	Proficient	Competent	Developing
Robot Design concepts	Can explain robot sensor components and how they work to others. Project demonstrates use of sensors in lesson and the best positions on the robot. Definitions of components are exhibited, by demonstration or in student documentation.	Demonstrates use of sensors on the robot. Full understanding of robot components is indicated to teacher.	Can make robot respond to sensors, but does not provide meaningful understanding of sensors. Basic understanding of hardware components demonstrated.	No evidence of understanding robot sensors and how they function. No understanding of hardware components demonstrated.
Software concepts	Complex command combinations and project goals demonstrated. Can explain command groups needed for using sensors, sensor position, and desired resulting robot position or action. Commands or groups of commands are demonstrated by presentation or documentation.	Student understands command groups needed for using sensors, sensor position, and desired resulting robot position or action. Commands or groups of commands mentioned in student's presentation.	When prompted, student has basic understanding of commands needed for using sensors, sensor position, and desired resulting robot position or action, but may not be presented in demonstration.	No evidence of understanding commands and how they function. No inclusion or mention of commands in student's presentation.
Coding concepts	Can debug code errors. Complex code combinations and unique use demonstrated. Can explain codes from most sections. Codes are included in student's demonstration or documentation.	Demonstrates understanding of default settings, loops, and conditional commands. Student references at least one of each command type code in their presentation.	When prompted, student has basic understanding of codes, but may not be mentioned in student's presentation.	No evidence of command codes and how they function. No inclusion or mention of codes in student's presentation.
Real-world concepts	Has innovative ideas to create real-world uses of robots, coding, and movement. Demonstrates understanding and documents it.	Understands some use of coding, movement, and robot uses in real-world applications. Includes at least one example in student's presentation.	Basic understanding of existence of coding, movement, and robot in real-world applications. May be explained verbally when prompted, but may not be included in presentation.	No evidence of understanding real-world applications using coding, movement, and robots.
Challenge Activity	Demonstrates object avoidance robot in action with clean, efficient code. Shows documentation of attempts, trials, and successful robot runs.	Uses conditional statements, loops, and distance sensor threshold settings correctly to avoid objects in the robot's path.	Can explain how a LiDAR sensor works, and explains the conditional statements with threshold settings necessary. May not have succeeded in a complete robot run without collisions.	Did not code an object avoidance robot. Does not express knowledge of how to code a robot to move with a distance sensor.

STANDARDS MAPPING

Common Core Standards

Common Core Standards link

Math skills can be reinforced when considering the distance to the targets, calculating those settings for the movement codes, and testing the results.

CSTA Standards for Students

https://csteachers.org/Page/standards

1B-AP-10

Create programs that include sequences, events, loops, and conditionals.

1B-AP-12

Modify, remix, or incorporate portions of an existing program into one's own work, to develop something new or add more advanced features.

1B-AP-15

Test and debug (identify and fix errors) a program or algorithm to ensure it runs as intended.

2-AP-10

Use flowcharts and/or pseudocode to address complex problems as algorithms.

2-AP-13

Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.

2-AP-17

Systematically test and refine programs using a range of test cases.

3A-AP-13

Create prototypes that use algorithms to solve computational problems by leveraging prior student knowledge and personal interests.

3A-AP-16

Design and iteratively develop computational artifacts for practical intent, personal expression, or to address a societal issue by using events to initiate instructions.

3A-AP-17

Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects.

3A-AP-22

Design and develop computational artifacts working in team roles using collaborative tools.

ISTE Standards for Students

ISTE Standards for students link

1.a, 1.b, 1.c, 1.d

Students leverage technology to take an active role in choosing, achieving, and demonstrating competency in their learning goals, informed by the learning sciences.

2.b, 2.d

Students engage in positive, safe, legal, and ethical behavior when using technology, including social interactions online or when using networked devices. Students manage their personal data to maintain digital privacy and security and are aware of data-collection technology used to track their navigation online.

3.a, 3.d

Students critically curate a variety of resources using digital tools to construct knowledge, produce

creative artifacts, and make meaningful learning experiences for themselves and others.

4.a, 4.b, 4.c, 4.d

Students use a variety of technologies within a design process to identify and solve problems by creating new, useful, or imaginative solutions. This includes creating innovative artifacts or solving authentic problems, considering design constraints and calculated risks, using prototypes as part of a cyclical design process, and developing capacity to work with open-ended problems.

5.a, 5.b, 5.c, 5.d

Students develop and employ strategies for understanding and solving problems in ways that leverage the power of technological methods to develop and test solutions.

6.b, 6.c, 6.d

Students communicate clearly and express themselves creatively for a variety of purposes using the platforms, tools, styles, formats, and digital media appropriate to their goals.

7.b, 7.c

Students use digital tools to broaden their perspectives and enrich their learning by collaborating with others and working effectively in teams locally and globally.

NCSS Standards

NCSS Standards link

NGSS Standards

NGSS Standards link

Interdisciplinary and 21st Century Connections

This lesson covers areas related to Coding/Computer Science.
21st Century Connections: Critical thinking • Creativity • Collaboration • Communication • Technology literacy • Flexibility • Leadership • Initiative • Problem solving • Mathematical calculations • Predictive reasoning

Modifications and Accommodations

Provide modifications and accommodations as appropriate based on student needs, IEP, 504, etc.

- Students can work in teams to integrate a paired programming approach
- Sample code can be provided for students to deconstruct and modify
- Provide adaptive mouse and/or trackball for navigation
- Provide larger monitor for ease of viewing the entire program
- Provide facilitated support to focus on appropriate software panels and reinforce nomenclature
- Add alternative measuring labels when working with the LiDAR sensor's thresholds

LESSON PLAN

LET'S TRAIN VIRTUAL ROBOTS

LESSON 4: COLLISION AVOIDANCE ROBOT

UNREAL ENGINE