HOUR OF CODE

UNREAL ENGINE

MOVING PLATFORMS AND CHECKPOINTS: LOOPS AND BOOLEAN VARIABLES IN UNREAL ENGINE

Lesson/Author/Class Information

Lesson Title: Loops and Boolean Variables in Unreal Engine

Content/Grade: Computer Science/Hour of Code™: Grades 8–12

Lesson Timeframe: One hour

Teacher Guide

Student Guide

Author Contact

Steve Isaacs teaches Game Design and Development as a quest or choice-based learning environment that provides students with opportunities to take different approaches to meet learning outcomes based on their own interests, in terms of content and project options.

Brian Dickman studied Computer Science and operates a full-time game development studio that produces entertaining and educational content inside popular video games.

Ian Southwell works at the Rocky Mountain College of Art and Design where he teaches in the Game Art and Animation department, as well as running the RMCAD Fabrication Laboratory. He also works as the Creative Director for Cleverlike Studios.

Email: stevei2071@gmail.com | brian@cleverlike.com | ian@cleverlike.com

Twitter: @mr_isaacs | @cleverlike | @IanSouthwell2

LinkedIn: https://www.linkedin.com/in/steve-isaacs/ | https://www.linkedin.com/in/cleverlike/
https://www.linkedin.com/in/southwellian/

Description of class/learning environment

This lesson is designed for Hour of Code™ during Computer Science Education Week. It can be used in any curricular area interested in participating in Hour of Code™. It can be used outside of Hour of Code™ in a game development or computer science course. It can be used as a stand-alone lesson, or in conjunction with the other activities to complete a larger project.

Lesson Overview

Many games include moving platforms to add a challenging and exciting game mechanic. Have you ever wondered how this is accomplished? Now it's your turn to become the developer and use **Unreal Engine**, one of the industry-standard tools for game development.

In this lesson, you will learn how to build a game level with floating islands, using moving platforms. To make it to safety, the player must time their jumps to traverse the sky, moving from island to island. You'll learn computer programming concepts including **loops** and **Boolean variables**, and how important they are, particularly in game development. This lesson is the second part of the Unreal Engine Hour of Code™ series.

DESIRED RESULTS

What are the learning outcomes for students?

Essential Questions/Big Ideas

- Can students learn computer science concepts as part of a meaningful activity rather than simply learning syntax as an isolated skill?
- Will learning computer science concepts, like loops and Boolean variables, through an activity in Unreal Engine create a general understanding of the concept in a coding environment?
- Can students learn computer science concepts through game mechanics?
- Will students show more motivation to learn computer science when the concepts are introduced in a game environment?

Learning Outcomes/Objectives

The student will be able to:

- Demonstrate an understanding of loops as a computer science concept.
- Demonstrate an understanding of Boolean variables.
- Apply the understanding of loops and Boolean variables in the context of a game.
- Create and modify a game level in a true game engine (Unreal Engine) that incorporates the application of loops and Boolean variables

LESSON PLAN

Learning Activities

How to Use the Unreal Engine in Hour of Code ™ Lessons

This series of lessons has been designed to introduce students to computer science concepts in the context of using the industry-standard game development tool, Unreal Engine.

Each lesson is set up as a stand-alone lesson to teach a single coding concept in the span of about an hour as part of the Hour of Code™ initiative. The lessons are presented to encourage students to work through them in linear order, as they acquire skills built on previous skills they learned.

The lessons also work together so that a student can complete all five lessons and create a game experience with five different levels, demonstrating the different concepts. The activities lend well to students working in small groups to leverage the idea of *pair programming*.

Each lesson is accompanied by a student guide and a teacher guide with notes for the educator to deliver the lesson and support students in the process.

Using Unreal Engine

It is expected that students have some experience with the Unreal Engine interface prior to starting. To facilitate teaching with Unreal Engine, educators can familiarize themselves with the tool and how to use it in the classroom, using a short course we have developed for this purpose. We encourage you to take the course and earn the badge!

Getting Ready for This Activity

If students completed the previous activity (Collision Detection in Unreal Engine), they have already completed the Lesson 1 activity and can continue. If not, they can enable Lesson 1 in the Unreal Engine project by making Lesson 1 visible, right-clicking on it, and choosing **Change Streaming Method > Always Loaded**. Directions on how to do this are available in the Lesson 2 Teacher Guide.

Hook

Have students launch and play the game.

Discuss the gameplay experience:

- How do you think the moving platforms are programmed?
- Did you encounter any issues or bugs in the current build?
- Can you identify where loops may be incorporated in terms of the moving platform?

In this level, students should notice that they must wait and time their jump to land on the moving platform so it can take them across the hallway.

Explain to students that, in this activity, they will be learning game development and programming concepts related to animation, loops, and Boolean variables.

Introduction to Loops

In computer science, a loop is a programming structure that repeats a sequence of instructions until a specific condition is met. Programmers use loops to cycle through values, add sums of numbers, repeat functions, and many other things.

– from https://techterms.com/definition/loop

For example, if you are on a track that is a quarter-mile in length and you want to run a mile, you would run around the track four times. This is an example of a loop. Once you have run around the track four times, you can stop.

Pseudocode is a way of writing coding concepts in a simple format that is easy for people to communicate and understand. The actual code in different programming languages will have different rules (or syntax), but pseudocode allows us to think about the code based on what we are trying to accomplish.

Here is an example of a loop written in pseudocode:

```
Repeat 4 Times {
        Run around the track
        }
```

This would be a great way for track coaches to program their athletes during practice.

Here are simple videos that explain how loops work in computer science.

- https://www.youtube.com/watch?v=WqmyVZnMWHY
- https://www.youtube.com/watch?v=BlXtMr7ge9Q

Loops can be used in any coding language, and in game engines like Unreal Engine. In the following activity, we will create and loop animation as an example of creating an infinite loop.

Introduction to Boolean Variables

The boolean data type is often used for making decisions. A boolean variable can only have two different values: true and false. There are different ways of comparing values. We can test if a number is greater or smaller than another number, if they are equal, if they are different. In each case a value of true or false is returned. We can print this value, or store it in a variable of type boolean.

– from https://funprogramming.org/94-Boolean-true-or-false.html

Here's an example of Boolean variables written in pseudocode:

```
If player collides with checkpoint Then {
        var checkpoint = true
        }
```

Here is a simple video that explains how Boolean expressions work in coding:

- https://youtu.be/y3rCKJNOwpA

Activity

In this lesson, you will be introduced to the concept of loops by converting a few simple floating islands into challenging moving platforms. You will also be using Boolean variables to add checkpoints that will reward players for completing difficult tasks. When reaching the checkpoint, the starting point for the player will be set to that location instead of the beginning of the game. This makes it less frustrating for the player as they can continue to try the level from the checkpoint.

Designing for a positive player experience

Students can make the course as easy or as difficult as they like. They should be mindful that difficult does not always equal fun. You might think your level is too easy to complete because you are the designer and have played through your level many times. Peers will playtest and provide feedback. Developers should be sure to take all the feedback from their playtesters seriously because this will make the game more enjoyable for a wider group of players. The more playtesters you work with, the more successful your game will become.

For step-by-step directions, provide students with the Student Guide and refer to the Teacher Guide to help guide students through the process.

External Resources

Your First Hour with Unreal Engine:
https://www.unrealengine.com/en-US/onlinelearning-courses/your-first-hour-with-unreal-engine

How Boolean Expressions Work in Coding (video):
https://www.youtube.com/watch?v=y3rCKJNOwpA&feature=youtu.be

ASSESSMENT

Assessments

Rubric
Hour of Code™ Moving Platforms and Checkpoints: Loops and Boolean Variables in Unreal Engine

	Developing	**Competent**	**Proficient**	**Distinguished**
Project Content and Learning Objectives	Project does not convey the required information or understanding as it pertains to the learning objectives.	Project shows a basic understanding of loops and Boolean variables.	Project reflects understanding of loops and Boolean variables in coding, and how looping can be accomplished with animation using a level sequence in Unreal Engine. Demonstrates understanding of conditional statements and Boolean variables.	Project reflects exemplary understanding and application of the level sequence to create looping animation that impacts gameplay. Mastery of the learning objectives is met or exceeded. Student effectively incorporates the use of checkpoints to demonstrate the use of conditional statements and Boolean variables.
Project Development and Functionality	Project does not work, or has major flaws that prevent its intended use.	Project demonstrates basic functionality, and has only minor flaws.	Project functions in the way the student intended and provides general guidance for the end user.	Project is functional and refined, with extra features that exceed the requirements.
Project Aesthetic and Design	Project requires more attention to the look and feel of the experience and the general design.	Project shows some attention to aesthetics and thoughtful design, but is incomplete or lacking in some aspects of layout and design.	Project is well organized and pleasing to the eye. The design makes sense in the context of the activity and creates a well-designed experience for the player.	Great attention to design. The environment is inviting, and provides the user with an engaging world to explore in order to experience the puzzle activities.
Reflection	Student demonstrates difficulty describing Loops and Boolean variables, and how they are represented in this activity.	Student describes the basics of loops and Boolean variables, and has a general understanding of how they relate to game development and programming.	Student provides a thoughtful reflection or explanation of loops and Boolean variables, and incorporates the concepts into practical use in Unreal Engine.	Student can eloquently explain the concepts of loops and Boolean expressions and how they can be represented using Unreal Engine.

Standards Mapping

CSTA Standards for Students: https://csteachers.org/Page/standards

1A-AP-09
Model the way programs store and manipulate data by using numbers or other symbols to represent information.

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.

3B-AP-22
Modify an existing program to add additional functionality and discuss intended and unintended implications (e.g., breaking other functionality).

Interdisciplinary and 21st-Century Connections

This lesson covers areas related to coding and Computer Science.
21st Century Connections:

- Critical thinking
- Creativity
- Collaboration
- Communication
- Technology literacy
- Flexibility
- Leadership
- Initiative
- Social skills

Modifications and Accommodations

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

Students can work in teams to integrate a paired programming approach

The completed project can be provided for students to deconstruct or modify.