UNREAL ENGINE

# REACH NEW HEIGHTS WITH POWER-UPS AND COLLECTIBLES: WORKING WITH PUBLIC VARIABLES IN UNREAL ENGINE

## Lesson/Author/Class Information

Lesson Title: Working with Public 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

In games, *power-ups* allow the player to perform actions they might not otherwise be able to perform. For example, you can add a speed boost to help a player reach a goal faster, or you can add a boost that makes your player invincible for a period of time. Oftentimes, power-ups are a function of changing a variable to provide this short term effect.

In this lesson, you will be adding a power-up that will allow the player to reach platforms that are much higher than they can currently reach. You will explore **Blueprints**, the visual scripting system in **Unreal Engine**, and make changes to the **variables** responsible for the desired outcome. Then you will add coin pickups to entice the player to jump to these out-of-the-way places.

Ready, Set, Jump!

## DESIRED RESULTS

### 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 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 the structure of Blueprints in Unreal Engine.
- Demonstrate an understanding of variables.
- Apply the understanding of modifying variables in the context of a game.
- Create and modify a game level in a true game engine (Unreal Engine) that incorporates changing variables.
- Work with variables to determine the correct value required to achieve a desired result.

# 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 (Loops and Boolean Variables in Unreal Engine), they already completed the Lesson 2 activity and can continue. If not, they can enable Lessons 1 and 2 in the Unreal Engine project by making Lessons them visible, right-clicking on them, and selecting **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 jump game mechanics work?
- Did you encounter any issues or bugs in the current build?
- Can you identify where variables can be incorporated in terms of the jump mechanic?

Students should notice that the next floating island is out of reach. They are unable to make the jump to that platform.

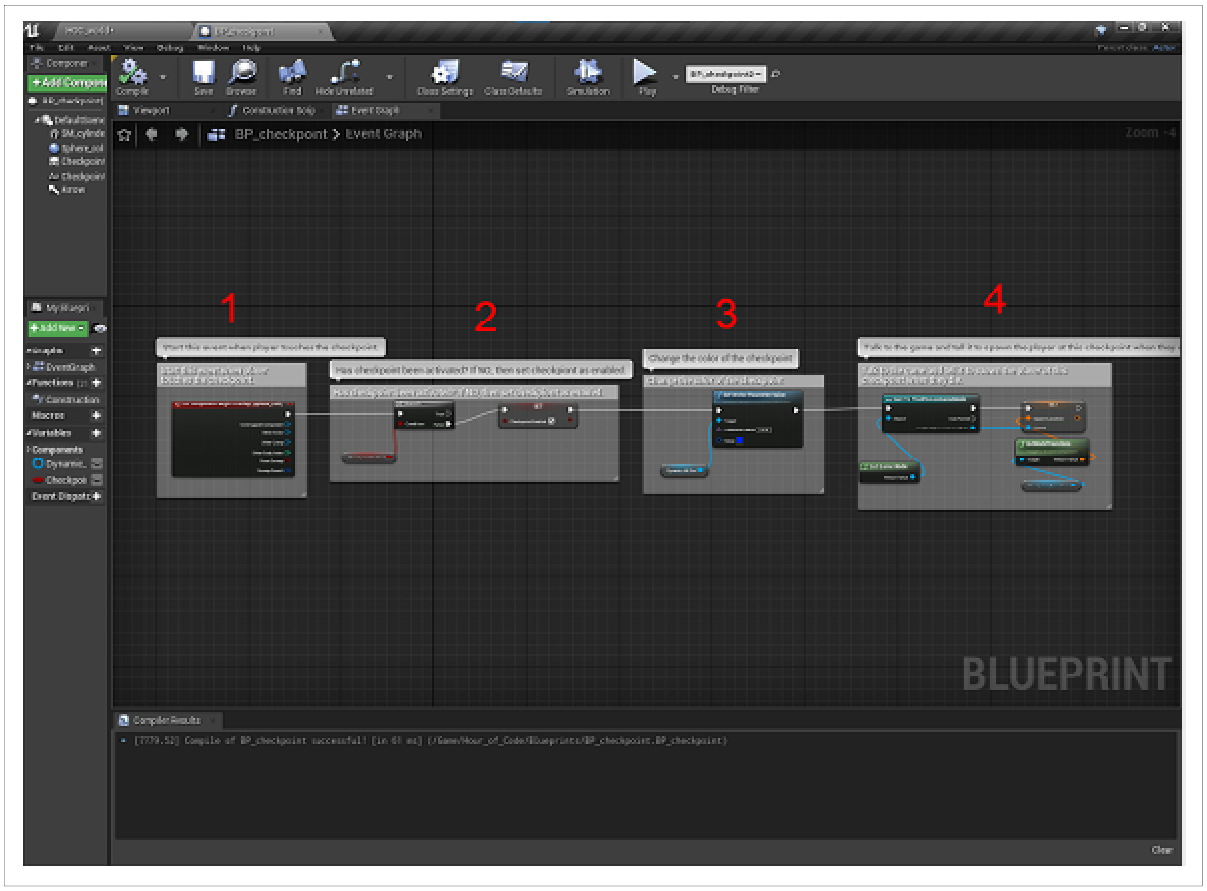
Explain to students that, in this activity, they will be learning game development and programming concepts related to variables and modifying the existing code.

## Introduction to Blueprints and Variables

**Blueprints.** The Blueprints Visual Scripting system in Unreal Engine is a complete gameplay scripting system based on the concept of using a node-based interface to create gameplay elements from within Unreal Editor. As with many common scripting languages, it is used to define object-oriented (OO) classes or objects in the engine. As you use UE4, you'll often find that objects defined using Blueprint are colloquially referred to as just "Blueprints."

This system is extremely flexible and powerful as it provides the ability for designers to use virtually the full range of concepts and tools generally only available to programmers. In addition, Blueprint-specific markup available in Unreal Engine's C++ implementation enables programmers to create baseline systems that can be extended by designers.

– from https://docs.unrealengine.com/en-US/Engine/Blueprints/GettingStarted/index.html



Start this event when player touches the checkpoint

Has checkpoint been activated? If NO, then set checkpoint as enabled

Change the color of the checkpoint

Talk to the game and tell it to spawn the player at this checkpoint when they

```
[7779.52] Compile of BP_checkpoint successful! [in 61 ms] (/Game/Hour_of_Code/Blueprints/BP_checkpoint.BP_checkpoint)
```

A Blueprint showing the drag-and-drop visual coding in Unreal Engine

**Variables.** In programming, a variable is a value that can change, depending on conditions or on information passed to the program. Typically, a program consists of instructions that tell the computer what to do and data that the program uses when it is running. The data consists of constants or fixed values that never change, and variable values (which are usually initialized to "0" or some default value because the actual values will be supplied by a program's user). Usually, both constants and variables are defined as certain data types. Each data type prescribes and limits the form of the data. Examples of data types include an integer expressed as a decimal number, or a string of text characters, usually limited in length.

– from WhatIs.com

For our purposes, think of a variable as a value, or a number, that can be changed at any time during the game to achieve a desired result. In our case, we want to temporarily change the jump velocity so the player can jump higher to reach the next platform.

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.

In terms of pseudocode, this could look like:

**Declare / set variables**

```
    var jump_velocity=1000
    cooldown=0
```

**Change velocity when player collides with speed_boost**

```
    Check Player for speed_boost
    Does player have speed_boost?
    If YES, jump_velocity= 2000
    cooldown=60
```

**Loop Until the cooldown is 0**

```
    Begin Loop
        Check Player for speed_boost
            Does player have speed_boost?
            If YES, add cooldown=cooldown-1
                Does cooldown=0
                    If YES, reset jump_velocity and cooldown
                    jump_velocity=1000
                    cooldown=0
    Loop Again IF the cooldown > 0
    Continue running until cooldown is 0
```

Here are several videos that explain variables in the context of coding:

*CS Principles: Intro to Variables Part 1:* https://youtu.be/G41G_PEWFjE
*CS Principles: Intro to Variables Part 2:* https://youtu.be/ijjVDBPwAlo

Variables can be used in any coding language, and in visual coding environments like Blueprints.

In the following activity, we will modify existing public variables to accomplish our task.

## Activity

In this lesson, you will be introduced to the Blueprints Visual Scripting in Unreal Engine and modify existing variables to create a power-up enabling the player to jump higher for a short amount of time by changing the velocity of the player and the time for a cool down associated with the power-up. Students will add power-ups where needed throughout the level and modify the variable values as needed. In addition, students will add collectibles to the coin that can be picked up by the player. This will be an introduction to collectibles. In a later activity, you will associate the score as a variable to the collected coins.

## Designing for a positive player experience

Students can make the course as easy or difficult as they like. They should be mindful that difficult does not 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 your play testers very seriously because this will make the game more enjoyable for a wider group of players. The more play testers 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

Introduction to Blueprints Visual Scripting:
https://docs.unrealengine.com/en-US/Engine/Blueprints/GettingStarted/index.html

# ASSESSMENT

## Assessments

### Rubric
### Hour of Code™ Reach New Heights with Power-Ups and Collectibles

| | 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 Blueprints and variables. | Project reflects understanding of the basic structure of Blueprints and how to change the value of variables to get the desired result in coding. | Project reflects exemplary understanding and application of Blueprints and how visual scripting works in Unreal Engine. Mastery of the learning objectives is met or exceeded. Student effectively incorporates the use of variables to get desired results and demonstrates this in multiple instances in the game. |
| **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 is intuitive 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 while experiencing the puzzle activities. |
| **Reflection** | Student demonstrates difficulty describing Blueprint visual scripting, variables, and how they are represented in this activity. | Student describes the basics of Blueprints and variables and has a general understanding of how they relate to game development and programming. | Student provides a thoughtful reflection or explanation of Blueprints and variables and incorporates the concepts into practical use in Unreal Engine. | Student can eloquently explain the concepts of Blueprints Visual Scripting and variables and how they are represented and modified 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-09**
Create programs that use variables to store and modify data.

**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-14**
Debug (identify and fix) errors in an algorithm or program that includes sequences and simple loops.

**2-AP-10**
Use flowcharts and/or pseudocode to address complex problems as algorithms.

**2-AP-11**
Create clearly named variables that represent different data types and perform operations on their values.

**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.