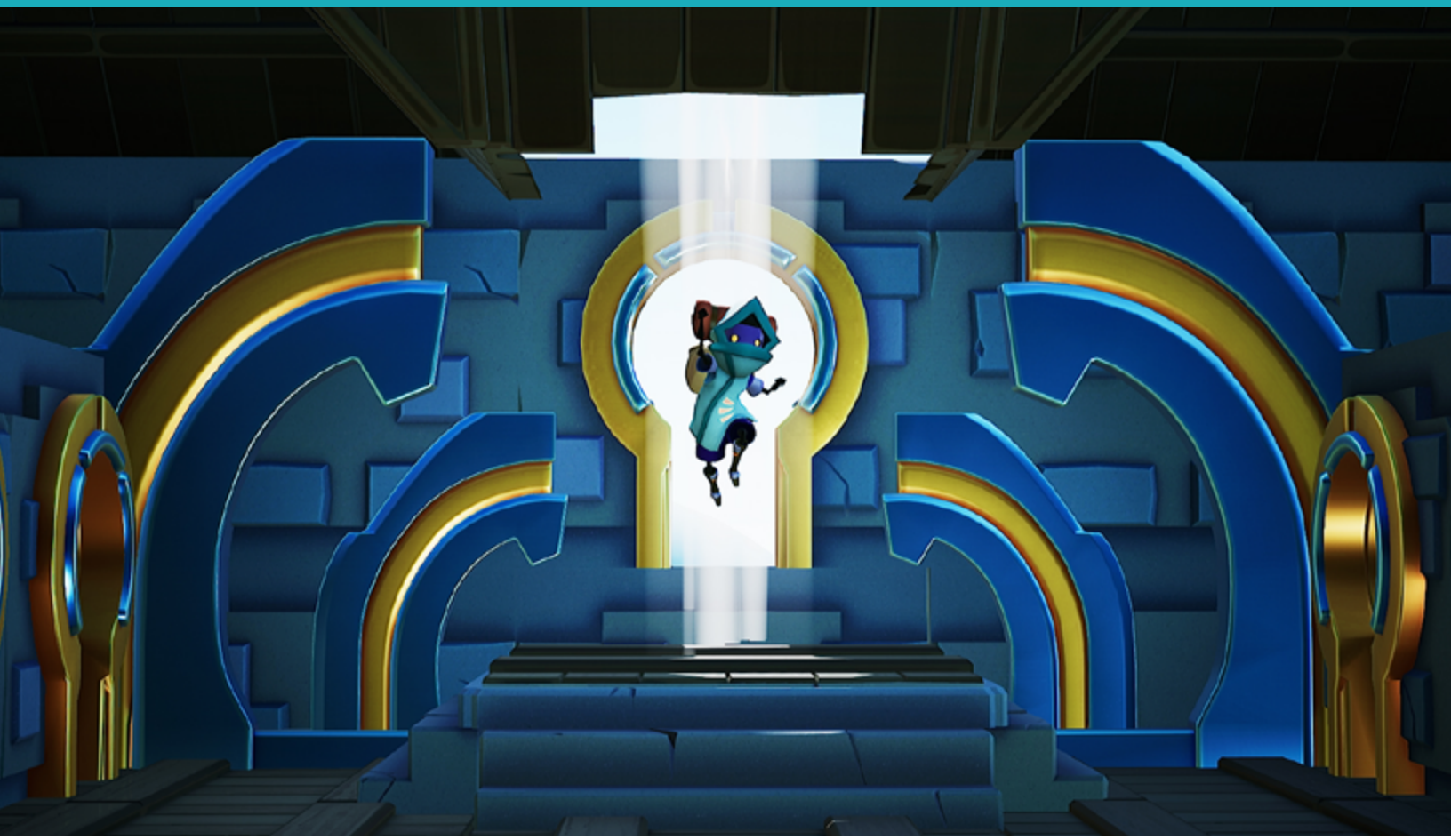
HOUR OF CODE



UNREAL ENGINE

# POLISH AND PUBLISH YOUR GAME:

## WORKING WITH WIDGETS AND THE HEADS-UP DISPLAY IN UNREAL ENGINE

## Lesson/Author/Class Information

Lesson Title: Working with Widgets and the Heads Up Display 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

Think about some of your favorite games. As a player, is the information displayed in a way that lets you know what's going on? Are you aware of your score or health? Can you easily view what items are in your inventory? If you have special abilities, is there a way for you to see whether they are currently available or in a cooldown period? These are all elements of the **User Interface (UI)** or **Heads-up Display (HUD)**. When implemented effectively, they add to the positive user experience in a game.

It's your turn to become the developer and use **Unreal Engine**, one of the industry-standard tools for game development, to add this functionality to your game. You will add UI and HUD elements to enhance the player experience. In addition, you will modify your game to make it your own. Finally, you will package (publish) your game so you can share it with the world!

# 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 related to the Heads-Up Display (HUD) 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 HUD and modify settings related to HUD elements.
- Demonstrate an understanding of how to incorporate widgets and work with them in the Blueprints Visual Scripting environment.
- Incorporate what they have learned to enhance a game level using Unreal Engine.
- Package and publish an Unreal Engine project so the end user can launch and play the game outside of the UE environment.

---

# 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 (Working with Conditional Statements in Unreal Engine), they already completed the Lesson 4 activity and can continue. If not, they can enable Lessons 1 to 4 in the Unreal Engine project by making them visible, right-clicking on them, and selecting **Change Streaming Method > Always Loaded**. Directions on how to do this are available in the Lesson 5 Teacher Guide.

At the end of this activity, each student will package and publish their game. Visual Studio must be installed on the computers being used. Ensure it is installed or ask your IT resource to verify that Visual Studio is installed on each computer.

The following will help facilitate the process:

Visual Studio installer: https://visualstudio.microsoft.com/downloads/

Packaging Your Game Tutorial:
https://docs.unrealengine.com/en-US/Programming/Development/VisualStudioSetup/index.html

## Hook

Have students add a key to their level, then launch and play the game.

Discuss the gameplay experience:

- Did you notice anything when you opened the door? Students may have noticed that an icon for a key showed up once you opened the door.
- Did you make it to the goal? If so, did you notice text including scoring information appeared on the screen? Do you know what onscreen text is referred to in games?

Explain to students that, in this activity, they will be learning how to work with the **Heads-up Display (HUD), Widgets**, and packaging their game as a stand-alone, executable file that others can open without needing to have Unreal Engine installed.

## Introduction to HUD

**HUD.** In video gaming, the heads-up display (HUD) or status bar is the method by which information is visually relayed to the player as part of a game's user interface. It takes its name from the head-up displays used in modern aircraft.

– from https://en.wikipedia.org/wiki/Heads-up_display_(video_games)

How the game communicates and interacts with the player is extremely important. **User Interfaces (UIs)** and **Heads-up Displays (HUDs)** are the game's way of providing information about the game to the player and, in some cases, allowing the player to interact with the game.

The **HUD** is the base object for displaying elements overlaid on the screen. Every human-controlled player in the game has their own instance of the AHUD class, which draws to their individual **Viewport**. In the case of splitscreen multiplayer games, multiple Viewports share the same screen, but each HUD still draws to its own Viewport. The type, or class, of HUD to use is specified by the game type being used.

– from https://docs.unrealengine.com/en-US/Gameplay/Framework/UIAndHUD/index.html

For example, when you are playing games, you may see menus, a health bar, lives, score, inventory items, etc. When these items are displayed on the screen, they are part of the HUD. Take a look at the wikipedia article above for more examples of other features that could be part of the HUD. Essentially, all text, icons, and other items displayed on the game screen related to the User Interface are part of the HUD.

Ask students: Can you think of some games you play that provide a lot of information to the player through the user interface or heads-up display?

**Accessibility.** Working with the UI and HUD provide opportunities to address accessibility issues through computer science.

Accessibility of user interfaces can be approached through usability. [...] Accessibility makes sure the user interface is designed to be effective, efficient, and satisfying for more people—especially people with disabilities, in more situations—including with assistive technologies.

– from http://www.uiaccess.com/accessucd/background.html

Have students read the article above and discuss ways they can be sensitive to the needs of all users when they develop a user interface.

## Introduction to Widgets

We will introduce a new type of Blueprint as well, the Widget blueprint. Think of a widget as a HUD element. They are displayed onscreen for the player, and because they are a type of blueprint, they can contain code that changes what they display. Remember the key icon that was displayed after you opened the door in Lesson 4? That icon is part of a widget that is told to display after the player uses the door.

## Activity

In this lesson, you will work with some aesthetic enhancements that will improve the user interface. You will work with the HUD to display information to the user that includes icons showing whether they have certain items or powerups; information to the player when they complete the game, including a message, time, number of items collected; and a way to exit the game. Refer to the ideas presented earlier regarding accessibility as you make any changes to the UI in your game.

In addition, you will have the opportunity to use what you've learned so far to modify the level to give it your own special touches. You can add islands and platforms to change the flow and feel of the game, place collectibles throughout the game, including some that might be harder to find to provide a challenge for the player, and carry out other personal changes to make the game your own.

Finally, you will learn how to package a game for distribution. This means that you will be able to publish (share) a version of your game with others in the form of an executable file that can be run as a stand-alone game without the end user needing to install Unreal Engine.

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

Heads-up Display (video games):
https://en.wikipedia.org/wiki/Heads-up_display_(video_games)

User Interface and HUDs in Unreal Engine:
https://docs.unrealengine.com/en-US/Gameplay/Framework/UIAndHUD/index.html

Accessibility in User-Centered Design:
http://www.uiaccess.com/accessucd/background.html

Widget Blueprints Documentation in Unreal Engine:
https://docs.unrealengine.com/en-US/Engine/UMG/UserGuide/WidgetBlueprints/index.html

Creating Widgets in Unreal Engine:
https://docs.unrealengine.com/en-US/InteractiveExperiences/UMG/UserGuide/CreatingWidgets/index.html

Visual Studio installer:
https://visualstudio.microsoft.com/downloads/

Packaging Your Game Tutorial:
https://docs.unrealengine.com/en-US/Programming/Development/VisualStudioSetup/index.html

Your First Hour with Unreal Motion Graphics (UMG):
https://www.unrealengine.com/en-US/onlinelearning-courses/your-first-hour-with-umg

# ASSESSMENT

## Assessments

### Rubric

### Hour of Code™: Polish and Publish Your Game: Working with Widgets and the Heads-up Display 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 heads-up display and widgets. | Project reflects understanding of HUD, widgets and how to package, publish, and distribute a game created with Unreal Engine. | Project reflects exemplary understanding of key concepts explored in this lesson. Mastery of the learning objectives is met or exceeded. Student incorporated skills learned to develop and publish an engaging game experience. |
| **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. Student enhanced the features and display of the HUD to further engage the player. |
| **Reflection** | Student demonstrates difficulty describing heads-up display and widgets and how they are represented in this activity. | Student describes the basics of HUDs and widgets and has a general understanding of how they relate to game development and programming. | Student provides a thoughtful reflection or explanation of how the HUD and widgets are incorporated to enhance the player experience. Student can describe how publishing works and how packaging a game as an executable file makes it easier to distribute. | Student can eloquently explain the concepts of HUDs, widgets, and packaging and publishing a game. Student clearly grasps all of the concepts presented in this lesson. |

## Standards Mapping

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

**1A-DA-06**
Collect and present the same data in various visual formats.

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

**1A-AP-10**
Develop programs with sequences and simple loops, to express ideas or address a problem.

**1A-AP-12**
Develop plans that describe a program's sequence of events, goals, and expected outcomes.

**1A-AP-14**
Debug (identify and fix) errors in an algorithm or program that includes sequences and simple loops

**1A-AP-15**
Using correct terminology, describe steps taken and choices made during the iterative process of program development.

**1B-AP-16**
Take on varying roles, with teacher guidance, when collaborating with peers during the design, implementation, and review stages of program development.

**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-IC-19**
Brainstorm ways to improve the accessibility and usability of technology products for the diverse needs and wants of users.

**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
- Technological 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.