OBJECT-ORIENTED PROGRAMMING
BUILDING A BLOCK-BASED WORLD

LESSON PLAN
In this lesson, students will develop a class hierarchy by applying inheritance to actors in an Unreal Engine Blueprint-based game. The main activity of the lesson will feature students building a hierarchy of blocks that can be placed in the world. The resulting game will be a block-building system with several different blocks. At the end of the lesson, students can expand the game using one of several suggestions or their own research.

By completing this lesson, students will demonstrate the object-oriented programming (OOP) paradigm of inheritance in a highly visual and interactive manner.

Students who have been studying Inheritance and Polymorphism in AP Computer Science (APCS) using Java language will construct parallels with Java programming and observe how an object hierarchy can be used to build an open-ended and efficient application.

Students who are studying APCS should have completed Chapter 9 of the CS Awesome (or similar) textbook before attempting this lesson: https://runestone.academy/runestone/books/published/csawesome/Unit9-Inheritance/toctree.html

NOTE: It is not necessary that students have been exposed to Java programming or the lesson mentioned above. However, doing so will provide some scaffolding and context for this lesson.
TABLE OF CONTENTS

04
LESSON | AUTHOR | CLASS INFORMATION

05
LESSON OVERVIEW

05
HOW TO USE THE BLOCK GAME LESSONS IN YOUR CLASS

07
ASSESSMENT

09
STANDARDS MAPPING
LESSON | AUTHOR | CLASS INFORMATION

LESSON INFORMATION

Lesson Title: Object-Oriented Programming with Unreal Engine

Content/Grade: Secondary students enrolled in a technical education program. Students may also be enrolled in Advanced Placement Computer Science (APCS)

Lesson Timeframe: Three 1.5-hour classroom sessions

AUTHOR CONTACT

Author, Organization/Role: John Wagner, MAT-T

Organization: North Country Career Center, Newport, VT 05855

Role: Teacher

Email: john.wagner@ncsuvt.org

LinkedIn: https://www.linkedin.com/in/john-wagner-009743/

DESCRIPTION OF CLASS/LEARNING ENVIRONMENT

Located in a regional high school tech center, the classroom includes 16 student PCs running Windows 10 Pro with dual 22” monitors, NVIDIA Quadro K2000 video cards, and 16 GB RAM. Students in the class have had several introductory lessons with Unreal Engine and are proficient with Blender, Photoshop, and other game creation engines such as GameMaker. Additionally, students are enrolled in AP Computer Science.

The class uses Google Classroom to distribute class materials and integrate collaborative Google documents / slides into the instruction.

The instructor’s screen is broadcast in real time to student computers using RealVNC.

To effectively participate in this lesson, students will need:

- A computer that can adequately run Unreal Engine. The lesson was written using Unreal Engine 4.24.3 but should work with current versions of the platform.
- A document-sharing system for students to work collaboratively.
- Google Classroom with Google Drive and Docs Students should have Epic Games accounts and have completed introductory Unreal Engine lessons to familiarize themselves with the environment.
LESSON OVERVIEW

In this lesson, students will develop a class hierarchy by applying inheritance to actors in an Unreal Engine Blueprint-based game. The main activity of the lesson will feature students building a hierarchy of blocks that can be placed in the world. The resulting game will be a block-building system with several different blocks. At the end of the lesson, students can expand the game using one of several suggestions or their own research.

By completing this lesson, students will demonstrate the object-oriented programming (OOP) paradigm of inheritance in a highly visual and interactive manner.

Students who have been studying Inheritance and Polymorphism in AP Computer Science (APCS) using Java language will construct parallels with Java programming and observe how an object hierarchy can be used to build an open-ended and efficient application.

Students who are studying APCS should have completed Chapter 9 of the CS Awesome (or similar) textbook before attempting this lesson: https://runestone.academy/runestone/books/published/cawesome/Unit9-Inheritance/toctree.html

NOTE: It is not necessary that students have been exposed to Java programming or the lesson mentioned above. However, doing so will provide some scaffolding and context for this lesson.

HOW TO USE THE BLOCK GAME LESSONS IN YOUR CLASS

The Block Game project was designed so that teachers can integrate the content into their curriculum in a variety of ways to accommodate different kinds of classrooms:

- **The Flipped Classroom Model:** At home each night, students watch one video module and follow along in Unreal Engine. The following day, the teacher uses classroom time to provide specific help or opportunities for project enrichment.
- **The Teacher-Only Model:** Instead of using the video modules, the teacher uses the Lessons and Activities document to lead students through creating the Block Game. Modules 6, 8, and 11, and the video course, can be played during class for helpful review.
- **The Self-Guided Model:** Students go through the video modules independently to complete the Block Game, with teacher support as needed. After completing the project, time is dedicated to implementing the Bonus Objectives and/or the students’ own modifications.

There is no “correct” way to run the Block Game unit, and there may be more ways than just the three we have listed here.

It is recommended that teachers go through the entire Block Game video course before teaching the content to others.
ESSENTIAL QUESTIONS/BIG IDEAS

• How does inheritance make software development more robust?
• How does using an object hierarchy make software development more extensible?

LEARNING OUTCOMES/OBJECTIVES

• Students will be able to develop an object-oriented hierarchy.
• Students will be able to identify areas in game development and general software development where inheritance will lead to a more generalized and robust solution.
• Students will be able to define attributes and behaviors in general terms, and provide concrete implementations when appropriate.

LEARNING ACTIVITIES

A complete set of lessons / objectives and a teacher script is included in the document Block Game ULK Lessons and Activities.

Videos that coincide with each lesson to accommodate remote learning or students who might miss a lab session are included. These videos are on the Unreal Online Learning portal and you can find them here.

The following is a suggested sequence for the lesson:

☐ Teachers should complete all lessons before assigning the project to students.
☐ Demonstrate the finished project. Point out that there are several block types, and that they all share similar functionality (they can be placed, walked on, removed), but that they also differ from each other (color, breakable/unbreakable, flowers, and so on).
☐ Discuss inheritance and its applicability to object-oriented programming. If teaching OOP with Java, point out the Java Library and its hierarchy as well as Unreal Engine and its class hierarchy.
☐ Discuss a system of objects, and have students come up with examples in the real world or in games they may have played.
☐ Deliver the lessons as in-class labs, or independently (with the videos). Instructors can give students read-only access to the Block Game ULK Lessons and Activities file if.

NOTE: A lot of information is covered here. The entire lesson should be spread across at least three class sessions. Students should be encouraged to expand on the game, there are several areas mentioned at the end of the lesson for students to research, as well as mastery objectives for the lesson.
RESOURCES

Your First Hour with Unreal Engine:
https://learn.unrealengine.com/course/2503277

Unreal Engine 4 Documentation 2.5: Child Blueprints

Unreal Engine 4 Documentation: Functions

CS Awesome: Inheritance
https://runestone.academy/runestone/books/published/csawesome/Unit9-Inheritance/toctree.html

ASSESSMENT

Throughout the lessons, students will reflect on their progress in a shared journal. Students should be encouraged to comment on their classmates’ reflections using the document commenting feature. Teachers will assess the students’ work on a per-lesson basis by evaluating their actual project (using whatever means of source code sharing they have) and the shared journal. Teachers should supply feedback using the rubric below.

These formative assessments form the basis for grading.

The teacher can use the included journal as a document shared with each student for this feedback. For example, using Google Classroom, the teacher would create a copy of the assessment document for each student and write in that copy.

Additional assessments can be utilized, such as video walkthroughs provided by students or classroom gallery walkthroughs, to showcase individualized customizations.

RUBRICS

All scaled sketches are to be completed on graph paper (supplied). Objectives for each lesson should be evaluated using the proficiency-based rubric below. Students who score a 2 or 1 should always be afforded the opportunity to update their project and demonstrate proficiency level of 3 or higher.
Each area should be evaluated for each lesson.

<table>
<thead>
<tr>
<th>CRITERIA</th>
<th>DEVELOPING — 1</th>
<th>COMPETENT — 2</th>
<th>PROFICIENT — 3</th>
<th>DISTINGUISHED— 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content</td>
<td>The student:</td>
<td>The student:</td>
<td>The student:</td>
<td>The student:</td>
</tr>
<tr>
<td></td>
<td>Accessed the instructional content and applied some of the instruction to their project.</td>
<td>Followed the instructional content and applied most of the instruction to their project.</td>
<td>Followed the instructional content and applied the instruction to their project.</td>
<td>Followed the instructional content and applied all of the steps to their project. Is able to demonstrate relevance of content in other areas.</td>
</tr>
<tr>
<td>Engagement</td>
<td>The student:</td>
<td>The student:</td>
<td>Required minimal assistance to successfully complete the unit. When challenged, investigated solutions and attempted to remedy problems.</td>
<td>Sought own resources when challenged and required no additional assistance to complete the unit.</td>
</tr>
<tr>
<td></td>
<td>Required direct instruction numerous times to complete the unit. Made no attempt to fix errors on their own.</td>
<td>Required assistance numerous times to complete the unit. Investigated solutions on their own but did not attempt to try solutions.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Objectives</td>
<td>Project did not meet the objectives of the lesson.</td>
<td>Project met some of the objectives of the lesson.</td>
<td>Project met the objectives of the lesson.</td>
<td>Project exceeded the objectives of the lesson. Demonstrated mastery by performing one or more of the mastery objectives.</td>
</tr>
<tr>
<td>Reflection</td>
<td>Reflection in the journal was minimal.</td>
<td>Reflection in the journal was observed and mostly relevant.</td>
<td>Project met the objectives of the lesson.</td>
<td>Reflection in the journal was relevant to the unit and demonstrated several areas of learning. An annotated screenshot was included that related to and reinforced the reflection.</td>
</tr>
</tbody>
</table>
STANDARDS MAPPING

Computer Science Teachers Association
[https://www.csteachers.org/page/standards]

1B-AP-10 Create programs that include sequences, events, loops, and conditionals.
2-AP-12 Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.
2-AP-14 Create procedures with parameters to organize code and make it easier to reuse.
3A-AP-14 Use lists to simplify solutions, generalizing computational problems instead of repeatedly using simple variables.
3B-AP-12 Compare and contrast fundamental data structures and their uses.
3B-AP-14 Construct solutions to problems using student-created components, such as procedures, modules, and/or objects.

Proficiency Standards from Computer Programming and Game Design, North Country Career Center

CCCP.2.1 Identify the X/Y/Z axes.
CCCP.2.3 Apply materials and textures to models.
CCCP.8.5 Produce 2D and 3D games with game engines.
CCCP.9.1 Apply Application Programming Interface (API) specifications to various problems.
CCCP.9.3 Use mathematical operators.
CCCP.9.4 Apply conditional statements to control program flow.
CCCP.9.5 Use iteration where appropriate.
CCCP.9.8 Model objects in software.
CCCP.9.9 Manipulate Array Lists.
CCCP.9.12 Demonstrate and utilize Inheritance.
INTERDISCIPLINARY AND 21ST CENTURY CONNECTIONS

There are many tie-ins with other disciplines that can be incorporated into this lesson:

- **Programming:** Throughout the lesson, tie-ins and references to Java are made. This lesson provides a nice way to reinforce that programming involves applying a series of algorithms to various data models, regardless of language. Unreal Engine Blueprints are real programming!
- **3D modeling:** Students can create their own Static Meshes for use in the game with other software such as Blender or Maya.
- **Texturing and materials:** Students can create their textures using Photoshop or GIMP, and build complex textures.

MODIFICATIONS AND ACCOMMODATIONS

To support differentiation, this lesson includes video versions of all activities. Students can attend lectures/labs and watch the videos at their own pace.

The lesson includes several open-ended opportunities to incorporate additional mastery skills for students who finish early.

Teachers can provide support for students who are gaining proficiency at a slower pace while also allowing students to proceed more quickly if desired.

The asynchronous aspect of the shared journal allows students to see and reflect on what their peers are working on. Students should be encouraged to ask questions in the journal and seek input.