A picture containing toy

Description automatically generated

**LESSON PLAN**

**Title:** Hour of Code 2021 (TimeCraft)

**Ages:** 7+

**Subject Areas:** Computer Science, Maths, Language Arts, History, Visual Arts, Science, Engineering

**21st century skills:**Creativity, Critical Thinking, Problem-Solving

**Supported languages:**  English (United States), English (UK), German, Spanish (Spain), Spanish (Mexico), French (France), French (Canada), Italian, Japanese, Korean, Portuguese (Brazil), Portuguese (Portugal), Russian, Chinese (Simplified), Chinese (Traditional), Turkish, Polish, Indonesian, Ukrainian, Greek, Bulgarian, Hungarian, Finnish, Danish, Swedish, Norwegian, Slovak, Czech, Dutch.

**Multiplayer/Single Player:** Single Player

**Level of experience with Minecraft required (educator):** Beginner

**Lesson description:**  Fun and creative tutorial introduction to computer science (CS). Travel through time and use block-based or Python code to solve mysterious mishaps in Earth’s timeline. Collect clues and use critical-thinking skills to deduce the culprit in this adventure through time. Students also explore the integration of computer science across careers and personal intererests.

**Total time required:** 1 hour

**Education Standards:**

|  |  |  |
| --- | --- | --- |
| Standards | | |
| United Kingdom | Australia | UK – England |
| I can model daily processes by creating and following algorithms to complete tasks  **1A-AP-08** | I can follow, describe and represent a sequence of steps and decisions (algorithms) needed to solve simple problems  **ACTDIP004** | I can create and debug simple programs  **Key Stage 1** |
| I can develop programs with sequences and simple loops, to express ideas or address a problem  **1A-AP-10** | I can define simple problems, and describe and follow a sequence of steps and decisions (algorithms) needed to solve them  **ACTDIP010** | I can understand what algorithms are, how they are implemented as programs on digital devices, and that programs execute by following precise and unambiguous instructions  **Key Stage 1** |
| Debug (identify and fix) errors in an algorithm or program that includes sequences and simple loops.  **1A-AP-14** | Design, modify and follow simple algorithms involving sequences of steps, branching and iteration (repetition)  **ACTDIP019** | I can use logical reasoning to predict the behaviour of simple programs  **Key Stage 1** |
| I can compare how people live and work before and after the implementation or adoption of new computing technology  **1A-IC-16** | Design algorithms represented diagrammatically and in English, and trace algorithms to predict output for a given input and to identify errors  **ACTDIP029** | I can use logical reasoning to explain how some simple algorithms work and to detect and correct errors in algorithms and programs  **Key Stage 2** |
| I can create programs that include sequences, events, loops and conditionals.  **1B-AP-10** | I can define simple problems and describe and follow a sequence of steps and decisions (algorithms) needed to solve them.  **ACTDIP01** | I can use sequence, selection and repetition in programs; work with variables and various forms of input and output  **Key Stage 2** |
| I can compare and refine multiple algorithms for the same task and determine which is the most appropriate  **1B-AP-08** | I can implement simple digital solutions as visual programs with algorithms involving branching (decisions) and user input  **ACTDIP011** | I can design, write and debug programs that accomplish specific goals, including controlling or simulating physical systems; solve problems by decomposing them into smaller parts  **Key Stage 2** |
| Test and debug (identify and fix errors) a program or algorithm to ensure it runs as intended.  **1B-AP-15** | I can implement digital solutions as simple visual programs involving branching, iteration (repetition) and user input  **ACTDIP020** | I can understand several key algorithms that reflect computational thinking; use logical reasoning to compare the utility of alternative algorithms for the same problem  **Key Stage 3** |
| I can discuss computing technologies that have changed the world, and express how those technologies have changed and are influenced by cultural practices  **1B-IC-20** | I can implement and modify programs with user interfaces involving branching, iteration and functions in a general-purpose programming language  **ACTDIP030** | I can use two or more programming languages, at least one of which is textual, to solve a variety of computational problems; make appropriate use of data structures; design and develop modular programs that use procedures or functions  **Key Stage 3** |
| I can use flowcharts and/or pseudocode to address complex problems as algorithms    **2-AP-10** | I can collect, access and present different types of data using simple software to create information and solve problems  **ACTDIP009** |  |
| I can design and iteratively develop programs that combine control structures, including nested loops and compound conditionals  **2-AP-12** |  |  |
| I can decompose problems and sub-problems into parts to facilitate the design, implementation and review of programs  **2-AP-13** |  |  |

**Teacher preparation:**

* Install Minecraft: Education Edition by visiting <https://aka.ms/HourofCode2021>
* Watch [the introductory video](https://aka.ms/HOC2021Trailer) to this year’s Hour of Code theme.
* Watch [the walk-through video](https://aka.ms/HOC2021Walkthrough) to get a better understanding of the lesson and learn how to navigate through the Minecraft world.
* Download the [Educator Guide](file:///C:\Users\laylahbulman\Downloads\aka.ms\HOC2021educatorguide), designed to provide you with all the information you need to help your students have an enjoyable and successful Hour of Code, including all external resources and suggested instructions for guided, independent and hybrid/distant-learning environments
* Use the student-facing [presentation slides](C:\\Users\\laylahbulman\\Downloads\\aka.ms\\HOC2021presentationslides) for step-by-step instructions
* Download a copy of the [coding solutions](https://aka.ms/HOC2021CodingSolutions).
* Do you have additional questions? Check out the [FAQ](https://aka.ms/HOC2021FAQ).

**Theme overview:**

As a computer scientist for the Institute of Major Time Errors, it’s your job to correct the mysterious Time Splits appearing in history and to find who (or what!) is causing them.

Will you help fix the Time Splits and save history using your coding superpowers?

Will you find out the truth about who, or what, is causing these crazy changes in history?

In your TimeCraft mission, you need to:

* Travel back to exciting moments in world history
* Code your Time Agent to fix the Time Splits
* Use the clues to identify the Culprit (who or what is causing the Time Splits)

Students may use block-based or Python code.

**Learning objectives:**

* Understand the importance and benefits of computer science in all aspects of life.
* Analyse and solve problems using algorithmic thinking and problem decomposition.
* Practise computer science concepts such as sequences, events, loops and debugging.
* Create coding solutions to successfully complete a task or solve a problem.
* Recognise the expanded career connections offered through computer science.

**Student activities:**

**Introductory Activities (5 minutes):**

1. Students reflect on the following three questions:

* What is computer science?
* How is computer science (or computer science skills) used in schools?
* How is computer science used in the workplace (in different jobs)?

1. Students review key glossary terms that will assist their understanding of the learning experience.

**Coding Activities (30-40 minutes)**

Students start their coding journey at the portal to the Ephemera One, the central spaceship of the Institute for Monitoring Time Errors. They are computer scientists eagerly awaited by TARRA, the artificial intelligence (AI) robot who manages the vessel and its Time Agents, robots who time travel to different points in history to fix Time Splits using code.

Students will be asked to select between Block or Python for coding activities, then they must select their Time Agent. We recommend that beginner coders start with Blocks.

After selecting their agent, students will complete two onboarding coding challenges.

ONBOARDING CODING CHALLENGES

Challenge 1: Call your Time Agent. Receive your TALK communication device from TARRA and call your Time Agent to your location.

Challenge 2: Agent Move. Move your Agent forward so that it stands on the green block.

TIMELINE CODING CHALLENGES

Students will be directed to the first challenge, Big Band Jazz, which be a guided coding experience. Students will have two hints to solve each coding activity, before it is solved for them. Each coding experience will be followed by a short clue search to identify the culprit.

After completing the Time Split 1: Big Band Jazz, students are teleported back to the main lobby to select another challenge. Students can select from any of the Time Splits, but it is recommended that they follow the Time Splits shown on the large screen above the Timeline.

|  |  |
| --- | --- |
| **Time Split 1**  Big Band Jazz  (This is the Guided Challenge) | The great jazz musician has lost his beloved trumpet and has replaced it with a kazoo! Code your Time Agent to get through the maze and retrieve the musician’s trumpet to save jazz. |
| **Time Split 2**  The Great Pyramids of Giza | The Great Pyramids are now cubes!  Code your Time Agent to help the designer create a secure structure that will last for thousands of years… the pyramid. |
| **Time Split 3**  Moon Mission | The astronauts need your help – use your Time Agent to complete a faith maze and deliver the calculations to help the astronauts land on the Moon! |
| **Time Split 4**  The Great Wall of China | The Great Wall isn’t actually great yet... In fact, it’s very short!  The hungry pandas keep eating the bamboo scaffolding, which now means that the workers cannot build the great and tall wall they wanted. Code a bamboo garden to divert the hungry pandas! |
| **Time Split 5**  The Mona Lisa | Mona Lisa is no longer smiling – in fact, she is very upset. The Time Culprit went back and trampled on Mona Lisa’s garden, causing her famous smile to be a frown. Code your Time Agent to cheer her up by planting more flowers. |
| **Time Split 6**  First Flights | The Culprit has filled the aeroplane runway with holes. Instead of having aeroplanes, we only have hot air balloons. Code your Time Agent to fix the runway so the inventors can make their first flight! |
| **Time Split 7**  First Computer Scientist | The first computer science program played a song, but the Culprit has ruined the code. Use your Time Agent to fix the code to make the music play! |
| **Time Split 8**  Human’s Best Friend | Human’s best friend is no longer the dog, but a bear! Use your Time Agent to befriend a pup and lead a pup back to the humans to become best friends again. |
| **Time Split 9**  Palaeontology Puzzle | Oh no! The Culprit went back in time and stole some fossils, causing the brachiosaurus to have a short neck. Use your Time Agent to help replace the stolen fossils so the palaeontologists can build the brachiosaurus with a long neck! |
| **Time Split 10**  Elements of Discovery | The culprit snuck into the science lab and hid some of the elements. Use your Time Agent to find the hidden elements so the scientist can make the great discovery! |

Upon completing three coding activities, students will have solved the first loop of the game and identified the culprit. They are able to extend their coding activities by selecting a new time agent and coding additional Time Splits.

**Assessment:**

* What was your favourite part of the Hour of Code?
* What was the most challenging part of the Hour of Code?
* How did you use computer science skills today?
* What is one new thing you learned today?
* Why is computer science important for all people?
* Would you like to try Minecraft: Education Edition again?

**Differentiation:**

In an effort to increase student access to the Hour of Code, there are three different learning experiences for students: in-class with a teacher facilitator, in-class as a self-guided experience or in a remote (virtual) learning capacity. Each of the learning experiences has different levels of teacher support and modification for student success and participation in the Hour of Code.

**Teacher-Facilitating (face to face)**

***This type of learning experience is going to provide students with the highest level of teacher support***. As the teacher, you will be providing explicit instruction and modelling before releasing students into independently working through the coding activities in the Hour of Code. You will utilise the [Hour of Code Presentation](https://aka.ms/HOC2021Presentation) to guide you and your students through starting game play, coding activities and lesson reflection.

**Student Self-Guided**

***This type of learning experience is going to provide students with the least amount of teacher support***. As the teacher, you will make sure that your students are able to log in to the Minecraft: Education Edition platform. You will provide students with the Hour of Code Presentation to assist them with their self-paced and self-guided experience for game play. The presentation will provide students with explicit directions on how to participate in the Hour of Code. You should also provide students with a copy of the Visual Glossary, which is located at the end of this Educator’s Guide. It is recommended that you provide these materials to students in a digital medium of your choice, or print off paper copies so they are readily available for student use.

**Remote-Learning Experience**

This type of learning experience can be facilitated synchronously or asynchronously.

***If you are having students participate in a synchronous (live) virtual session***, you will facilitate the session just like the in-class experience (with teacher facilitation). As the teacher, you will be providing explicit instruction and modelling before releasing students into independently working through the coding activities in the Hour of Code. You will utilise the Hour of Code Presentation to guide you and your students through starting game play, coding activities and lesson reflection. Make sure that your virtual communication platform (like Microsoft Teams) is ready to share content and the sound is enabled.

***If you are having students participate in an asynchronous (i.e. not live) virtual session,*** you will facilitate the session just like the self-guided in-class experience. As the teacher, you will make sure that your students have access to the Minecraft: Education Edition platform prior to lesson. You will also need to provide students with the Hour of Code Presentation to assist them with their self-paced and self-guided experience for game play. The presentation will provide students with explicit directions on how to participate in the Hour of Code. You should also provide students with a copy of the Visual Glossary, which is located at the end of this Educator’s Guide. It is recommended that you provide these materials (presentation and glossary) to students in a digital medium of your choice (examples: Microsoft Teams channel, OneNote notebook, your learning management system, etc.).

If you plan to offer this lesson via remote learning, please consider these tips:

1. This lesson is not designed as a multiplayer experience. Each individual student should work in their own version of the world.

2. Divide students into pairs or small groups via breakout rooms so they can help one another troubleshoot as they solve the coding activities.

3. Given that students are likely to have varying levels of familiarity with Minecraft and coding, it may help to assign student group leaders to help peers who may get stuck.

For more information on remote learning in Minecraft: Education Edition, please refer to <https://aka.ms/remote-learning-kit>