**Storm survivor: Input, decision-making and loops**

**Years 5–6**

Please refer to the online lesson plan on the DT Hub to access all website links and additional resources.

**Summary**

Students use a visual programming language to create a game or quiz to help members of a community prepare for a severe weather event.

**Suggested steps**

Provide students with stimulus images of the after effects of a cyclone, storm or flood. Discuss the impact of severe weather events and natural disasters within a community. Invite students to share their own experiences. Use focus questions to strengthen students’ understanding; for example:

* How do people stay safe during a severe weather event (for example, a flood or cyclone)?
* What are some of the ways people can prepare for a severe weather event? (Answers might include securing loose outdoor items, moving cars undercover, cleaning out roof gutters and putting together an emergency kit.)
* How do emergency services inform the public that a severe weather event is imminent?
* How can social media be used to help the community stay informed?
* What websites or apps provide the public with information about severe weather events? (Answers might include the Bureau of Meteorology.)

**Optional activities: ICT Capability**

Students could:

* create short messages that could be posted on Twitter, Facebook or Instagram to alert the public about a severe weather event
* design the user interface for an app or website to help people prepare for severe weather events or natural disasters.

**Design and create a game or quiz**

Working collaboratively, students design and create a game or quiz to help others prepare for a severe weather event. Students must include algorithms involving user input, decision-making statements and loops, and must use data such as images and sound.

Students consider how a simple game or quiz could help people prepare for a severe storm, flood or cyclone. Brainstorm the different types of games or quizzes that could be created. For example:

* a game where players create an emergency kit by moving through a maze, avoiding obstacles or barriers, and gaining points for selecting appropriate emergency items along the way
* a game where players are rewarded for correctly preparing their home and surroundings prior to a cyclone
* a game where players choose between useful and non-useful items prior to a flood.

Before students create the game or quiz, ask them to identify algorithms used in online game action including branching (for example, needing to complete one task before moving on to another) and repetition (for example, obstacles moving back and forth).

Model the If-then and If-else code blocks in a visual programming environment such as Scratch.

    

Encourage students to consider how these concepts can be used in their game; for example:

* IF a player selects correct items from a list THEN they score points
* IF a player completes a task incorrectly THEN they have to start again.

The table below has examples of questions and ideas for consideration prior to students commencing the task.

|  |  |
| --- | --- |
| **Computational thinking** | Example questions:* What is the purpose of the game?
* What is the problem or issue?

Example answer: This game is being created to help people learn how to put together an emergency kit in preparation for a cyclone, flood or severe storm. More example questions:* What are the requirements for the game?
* How will you design algorithms to process user input (eg how will the user move their character in the game – with the arrow keys, the space bar)?
* What decision-making statements will be needed?
* How will you design algorithms for game actions involving branching? (eg when the player avatar selects a correct item for an emergency kit – they might then say ‘yippee’ and move on, or the player might score a point)
* How will you design an algorithm for user interactions in a game? (eg when the Up arrow is pressed, move the player avatar 10 steps upwards)

 |
| **Systems thinking** | Ask students to draw the screen layout and identify the items players will see when they are playing the game. Students will need to understand how digital systems process input and display output. Examples of user input include keystrokes and mouse clicks.As students design the user interface, ask them to consider:* What will the user interface look like?
* Will there be a background?
* What features of familiar user interfaces enable user interaction? (Answers might include input boxes, menus on web pages and on apps, and symbols to represent help, go back, home and volume.)
* How can important information on a user interface be made easy to find?
* How will players interact with the game? Will they be represented by a sprite or avatar?
* How will the players know the score? What actions are triggered by user inputs?
 |
| **Data** | What types of data are required? The types of data will be dependent on the game (eg images of torches, first aid kits, non-perishable items, water bottles; sounds; instructions for playing the game). |

**Discussion**

Prior to playing each other’s games ask students to independently review their own game and provide feedback, considering the following:

* How did you feel about thealgorithm design process?
* What were some of the challenges when completing the task?
* What instructions did you include in your code that required looping?
* What algorithms did you use that made your program more efficient?
* Did the game meet the intended purpose?
* Explain how you could improve the game to match the need or make the game more fun to play.
* What are different ways of extending and increasing the difficulty of the game?
* How could the game be adapted or modified to suit learners of a different age group?

Provide time for students to play each other’s games and quizzes. Initiate a whole class discussion to determine the success or limitations of each game.

Suggestions for feedback include:

* Were the instructions easy to follow?
* Was the game fun to play?
* Do you think the game helps people learn about preparing for a severe weather event?
* Was the user interface effective? In what way?
* How could the game be improved?

**Why is this relevant?**

In years 5 and 6, students should have opportunities to incorporate decision-making, repetition and user interface into their designs and implement their digital solutions, including a visual program.

Creating a digital solution usually starts with a problem that needs to be solved. This activity supports students to understand that they can create a digital solution that meets a local community need. It also helps them to incorporate decision-making and repetition into their program through the use of looping.

Looping is an important aspect of programming as it helps to achieve a desired outcome more efficiently. The order of instructions within a program is also important. While these instructions may be sequential or linear, through the use of decision-making statements the order of the program can be altered. For example, a loop is something that will continue to repeat until a certain condition is met (ie until a certain condition becomes true). In visual programming languages such as Scratch the loop blocks include the Repeat block and decision-making statements If-then and If-then-else*.* Here are some examples.

|  |  |  |
| --- | --- | --- |
| **Screenshot** | **Command** | **Explanation** |
|  | Repeat | **This loop will repeat the commands within it a specific number of times.**  |
|  | Repeat 10 times: turn 15 degrees clockwise | **This program is instructing a sprite to turn 15 degrees in a clockwise direction 10 times. The instructions contained within the Repeat block are looped until the condition ‘repeat this instruction a total of 10 times’ is met.**  |
|  | If-then | **The If-then block executes the instructions contained within it if the specified condition is met.****For example:** |
|  | If-then-else | **The If-then block executes the instructions contained in the top section if the specified condition is met. If the specified condition is not met then the instructions contained in the bottom section will be executed instead.** |

**Assessment**

Evaluate students’ understanding by observing processes undertaken in completing set tasks (for example, did students adapt the steps taken to complete tasks after consideration of more efficient processes?)

Use criteria to determine students’ ability to:

* create a simple game matching the features of the game to the purpose
* create and modify algorithms for user input, branching and repetition
* collect different types of data including images and sound
* use If blocks and Repeat blocks in a visual programming language.

**Curriculum links**

**Links with the Digital Technologies curriculum area**

|  |  |
| --- | --- |
| **Strand** | **Content description** |
| Processes and Production Skills | Design algorithms involving multiple alternatives (branching) and iteration (AC9TDI6P02) |
| Processes and Production Skills | Implement algorithms as visual programs involving control structures, variables and input (AC9TDI6P05) |
| Processes and Production Skills | Evaluate existing and student solutions against the design criteria and user stories and their broader community impact (AC9TDI6P06) |

**Links with other curriculum areas**

|  |  |
| --- | --- |
| **Strand** | **Content description** |
| HASS (Geography) | The management of Australian environments, including managing severe weather events such as bushfires, floods, droughts or cyclones, and their consequences (AC9HS5K05) |