# STEMin

# LESSON PLAN

## Coding a simple game in Scratch

In this activity, students will use Scratch to create a simple game, such as a maze or a character that catches objects.

**45-60 min** **Duration**

## Recommended age for this game

**10-12 years**

## Learning Objectives

<CODE/>

- Understand the basics of programming logic, including loops, sequences, and conditional statements.
- Develop problem-solving and critical thinking skills through coding.
- Create and test a functional game prototype.
- Foster creativity and collaboration in a digital environment.

# STEM in

## Materials and tools needed

- Computers or tablets with internet access.
- Scratch accounts for students (https://scratch.mit.edu).
- Projector or screen for teacher demonstrations.
- Headphones (optional, for game sound effects).

## Guidance for Teachers

### Activity description

Students will create a simple Scratch game, such as a maze or a catching game, using drag-and-drop coding blocks.

They will design the characters, set game rules, and test the game for functionality.

# STEM in

## Guidance for Teachers

### Preparation

- Familiarize yourself with Scratch's interface and create a simple example to demonstrate in class.
- Prepare a step-by-step guide or slideshow to introduce Scratch and the project objectives.
- Divide students into small groups if they'll work collaboratively.
- Test all devices to ensure compatibility with Scratch.
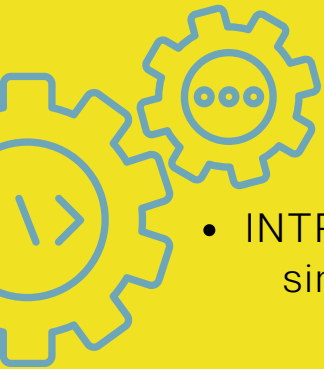
### Implementation steps

- INTRO: Explain what Scratch is and show a quick demo of a simple game. Highlight also key coding blocks like motion, events, loops, and conditions.

- PLANNING: Guide students to brainstorm game ideas and sketch their designs on paper. Encourage them to define game objectives, such as winning conditions or obstacles.

- CODING: Help students set up their Scratch projects and start coding. Circulate to support troubleshooting and encourage experimentation with blocks.

# STEM in

## Guidance for Teachers

- TESTING: Have students test their games, identify bugs, and improve them. Encourage peer feedback to refine their projects.

- SHOWCASE: Allow students to present their games to the class and explain their coding process.

### Follow-up and reflection

- Organize a "game fair" where students play each other's games and provide feedback.

- Assign a short reflection task:

## What did they enjoy?

## What was challenging?
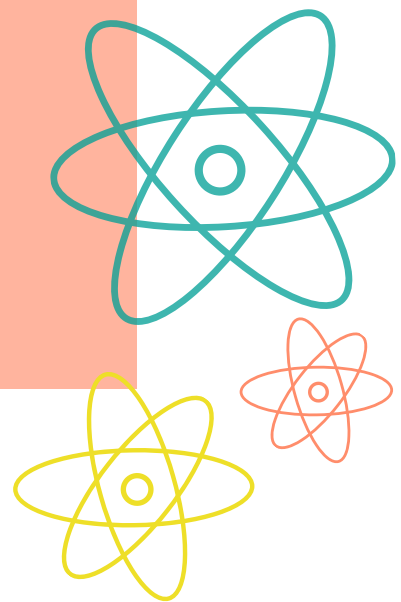
# Student Activities

| Activity description | Expected outcome | Technology integration |
|---|---|---|
| **Design a Game Plan** | Students will define their game's objective, rules, and characters. | Use Scratch's "backdrop" and "sprite" options to visualize designs. |
| **Code Game Movements** | Students will program sprites to move or interact with each other. | Utilize "motion" and "event" blocks in Scratch. |
| **Add Game Logic (Win/Loss)** | Students will use conditional statements to define outcomes. | Use "if-then" and "broadcast" blocks for logic implementation. |
| **Test and Debug** | Students will identify and fix coding errors. | Debug using Scratch's stage preview and block highlighting. |
| **Share Game** | Students will share projects and discuss their coding choices. | Use Scratch's sharing feature or class projector |

# STEM in

## Reflective questions for students

- What challenges did you face while coding your game, and how did you overcome them?
- Which coding blocks were the most useful for creating your game?
- If you could add one new feature, what would it be and why?
- How did you use feedback from others to improve your game?
- What did you learn about teamwork or problem-solving through this activity?

# STEM in

# Differentiation ideas

## Advanced Students

- Encourage them to add complexity, such as:
  Multiple levels or timers.
  Advanced logic like scoring systems.
  Additional animations or sound effects.
- Ask them to assist peers who need help, reinforcing their knowledge.

## Students with special needs

- Pair them with a buddy for additional support.
- Break the task into smaller steps and offer visual guides.
- Focus on celebrating progress over perfection.

# Tips

- Keep instructions simple and use visuals to explain coding blocks.
- Encourage creativity by allowing students to personalize their games.
- Allocate extra time for troubleshooting and debugging.
- Foster a supportive environment where students feel comfortable asking for help.
- Use peer feedback sessions to encourage collaboration and idea-sharing.

# STEM in
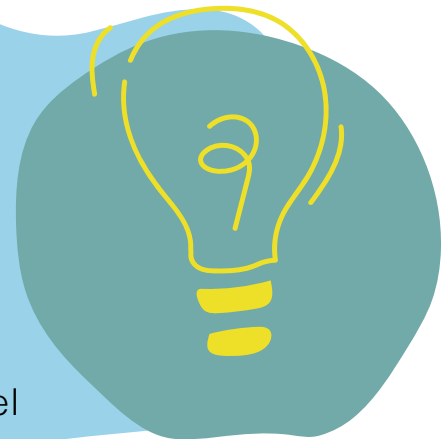
## Additional materials and references

- Scratch Website: https://scratch.mit.edu

- Getting started with Scratch Guide

- Interactive Coding Tutorials

- Video tutorial on How to use Scratch for Kids

VILLAFUENTE
ASOCIACIÓN

ALYTAUS VIDZGIRIO
MOKYKLA
1977

nekser

CC BY-SA 4.0 DEED