

Create a **fully playable HTML game** that helps undergraduate college students practice applying a concept or content they encountered in lecture earlier that day.

**This game should be about: [INSERT LECTURE TOPIC HERE]**

---

### Learning Goal

Design the game so that students **use and apply the material**, rather than simply recall definitions.

Before building the game, silently identify:

- 3–5 essential ideas students should understand about the topic.
- 3–5 common misconceptions or incorrect approaches students might have.
- The kinds of decisions, judgments, comparisons, calculations, classifications, or actions a student should be able to make after learning the material.

Use these insights to design the gameplay.

**Do not show this planning process to the student.**

### Game Design

Choose a game format that naturally fits the topic and allows students to practice the material through gameplay.

The game must be:

- **A real game, not a quiz disguised as a game.**
- Playable with minimal or no instructions.
- Understandable within the first 20–30 seconds.
- Designed to take approximately **3–5 minutes** to complete.
- Focused on one clear gameplay loop rather than many complicated mechanics.
- Interactive enough that students make meaningful decisions repeatedly.
- Designed around the topic rather than simply placing quiz questions inside a generic game.

Quiz-style questions may appear occasionally, but **the majority of the student's interaction should involve playing the game and applying the concept.**

## **Use misconceptions as game mechanics**

Common misconceptions should appear naturally in the game.

For example, misconceptions might become:

- Incorrect items the player must avoid.
- Tempting but flawed choices.
- Dangerous paths.
- False clues.
- Incorrect combinations.
- Ineffective strategies.
- Enemies or obstacles.
- Resources that must be used correctly.
- Decisions with consequences.

A student should be able to demonstrate understanding through their gameplay—not merely by answering multiple-choice questions.

## **Variety**

Choose the game genre that best fits the topic.

**Do not default to the same genre every time.** Vary the underlying gameplay across different requests whenever possible.

Potential genres include:

- Endless runner
- Collect-a-thon
- Maze exploration
- Obstacle course
- Treasure hunt
- Racing game
- Kart racer
- Hovercraft racer

- Survival arena
- Boss battle
- Rail shooter
- Tower defense
- Resource management
- Factory automation
- Restaurant simulator
- Detective mystery
- Courtroom trial
- Negotiation game
- Conveyor-belt sorting
- Assembly line
- Timeline builder
- Pattern detective
- Sokoban-style puzzle
- Match-3 puzzle
- Pipe-connection puzzle
- Memory game
- Rhythm game
- Dungeon crawl
- Mini-golf
- Marble roller
- Capture-the-flag

You may invent a different game structure if it fits the topic better.

**Prioritize instructional fit over novelty.**

**N64-Inspired Presentation**

Give the game a visual and audio style inspired by **late-1990s Nintendo 64-era 3D games**:

- Low-poly 3D-style visuals.
- Chunky geometry.
- Bold colors.
- Simple textures.
- Playful arcade-like presentation.
- Strong environmental shapes and landmarks.
- Retro HUD elements.
- Satisfying visual feedback for success and failure.

The game should feel like a **small, polished N64-era game**, not a modern educational quiz with a retro skin.

Do not use copyrighted Nintendo characters, artwork, logos, music, or other proprietary assets.

### **Feedback and Learning**

Give the player immediate, understandable feedback.

When a student makes a mistake:

- Clearly indicate what happened.
- Briefly explain the relevant concept.
- Connect the feedback to the student's action.
- Allow the student to continue playing whenever practical.

Do not punish mistakes so severely that students cannot finish the game.

Where appropriate, make the consequences of a misconception visible through the game itself.

### **Difficulty and Progression**

Create a simple progression:

1. Introduce the mechanic safely.
2. Give the student several opportunities to practice.

3. Introduce more challenging situations.
4. End with a short challenge that requires combining or applying what they practiced.

The game should feel progressively more difficult without becoming frustrating.

## **Accessibility**

The final game must be designed to meet **WCAG 2.1 accessibility principles** as closely as reasonably possible.

At minimum:

- All important information must be available to screen readers.
- Do not communicate essential information through color alone.
- Provide keyboard controls.
- Make interactive elements keyboard accessible.
- Provide clear focus indicators.
- Use sufficient color contrast.
- Provide text alternatives for important visual information.
- Use semantic HTML where appropriate.
- Avoid flashing or rapidly blinking content.
- Provide instructions in accessible text.
- Make feedback available in text, not only through animation or sound.
- Ensure the game remains usable without audio.
- Include appropriate ARIA labels where necessary.
- Use readable font sizes and clear visual hierarchy.

If a visual 3D-style mechanic cannot be made fully screen-reader accessible, provide an equivalent accessible interaction rather than making the mechanic inaccessible.

## **Technical Requirements**

Build the game as a **single self-contained HTML file**.

It should:

- Run by opening the HTML file in a modern web browser.

- Require no server.
- Require no login.
- Require no external database.
- Require no gradebook integration.
- Require no account.
- Avoid external dependencies when practical.
- Work without an internet connection once the file is downloaded.
- Include all CSS and JavaScript needed to play.
- Start immediately from the HTML file.

Do not ask the instructor to configure code, install packages, or modify anything other than the **GAME TOPIC** line.

### **Classroom Context**

Assume:

- Students are undergraduate college students.
- They encountered the topic in lecture earlier the same day.
- The game is for reinforcement and engagement, not formal assessment.
- Students may be playing individually on a laptop.
- The instructor may project the game or distribute the HTML file.
- Students should be able to understand the objective quickly without instructor intervention.

### **Content Accuracy**

Do not invent facts about the topic.

Use the standard academic understanding of the subject. If the topic has multiple conventions or interpretations, use the most broadly accepted undergraduate-level treatment.

Avoid introducing advanced material that students would not reasonably need to know to play the game.

## Final Deliverable

Return the **complete playable HTML file**.

Do not provide pseudocode, an outline, or a partial implementation.

The finished game should be immediately playable when saved as an .html file and opened in a browser.

Before finishing, internally check:

- Is this actually a game rather than a quiz?
- Does the gameplay require applying the target concept?
- Are common misconceptions incorporated into gameplay?
- Can a student finish in approximately 3–5 minutes?
- Is the game understandable within 30 seconds?
- Does the game work with keyboard controls?
- Is important information accessible to screen readers?
- Is the game visually and mechanically reminiscent of an N64-era game?
- Does the game work as a standalone HTML file?
- Would a college instructor realistically want to use this in class?

If any answer is no, improve the implementation before returning it.