Senior Unity Game Developer Test

About the Test

Welcome to the **Plan A Senior Game Developer Unity Practical Test**.
This is a fun way to test your abilities and familiarize you with some of the common challenges we face every day in our projects.

Here are some things you need to know before you start:

- The test consists of 3 tasks with increasing difficulty. Your mission is to complete them within the next 3 hours (don't worry, they won't take all of that time).
- The tasks are intentionally progressive in difficulty. Don't worry if you can't finish them all - any progress will be considered.
- Some gaps are left intentionally. If you find them, solve them in the way you think provides the best player experience.
- Please comment on anything you consider relevant, such as how something works, why you made a particular decision, or why you chose a specific data structure.
- We will evaluate not only the code itself but also its quality, your knowledge of the language and the engine, overall organization, and good separation of concerns.
- There are no absolute "correct" answers, but your solutions will be evaluated against maintainability and best practices.
- Don't worry too much about details - just solve the tasks as you normally would.

What You Will Deliver

At the end of the test, we expect an **email from you with a link to a GitHub repository** containing all the project files.

The game **must be executable** from the main scene without requiring extra setup.

Entries after 3h will not be accepted.

The Game

A simple puzzle game: a **6x5 grid** filled with colorful blocks.

Gameplay

- The player starts with **5 moves**.

- When they tap a block, that block and all connected blocks of the same color (up, down, left, right) are collected.
- Points are awarded based on the number of blocks collected (+1 for 1 block, +2 for 2, etc.).
- Empty cells are filled by blocks falling from above, and new random blocks are generated to fill the empty spaces.
- The game ends when the player runs out of moves.

Technical Requirements

- Mobile only, portrait mode.
- **No animations required** (gravity can be instant).
- Use **TextMeshPro** for text.

Instructions

- Deliver everything in a **single Unity project** under git version control.
- Focus on working solutions while showing clean, maintainable code.
- Commit as often as you like. After finishing each task, tag it as `question-1`, `question-2`, or `question-3`.
- If you had to make compromises due to time or unclear requirements, explain them in comments.

Task 1

Topic: Asset Integration

Scenario

You've just received assets from the art team, along with preview images. Your mission: integrate them into the game as close to the previews as possible, keeping the limitations of the project in mind.

Task

- Create a new 2D Unity project.
- Integrate all given assets, setting them up properly (fonts, 9-sliced sprites, etc.).
- Build the main scene (grid can be empty for now).
- Create a **Game Over** screen that matches the preview.

Expected Result

A single scene that resembles the preview and a Game Over screen object that also matches the preview.

Task 2

Topic: Data Handling

Scenario

You are developing the core UI for a match-3 puzzle game. Players need to see their current score and the number of moves remaining.

Task

- Implement a C# script to manage score and moves.
- Update the UI Texts created in Task 1 with these values.
- Simulate gameplay:
  - Start with 5 moves and a score of 0.
  - Add a "Make Move" button that decreases moves by 1 and increases score by 10.
  - When moves reach 0, show the Game Over screen and stop gameplay.
  - Add a "Replay" button that resets the game to the initial state.

Expected Result

A functional Unity scene showing score and moves, with a button to simulate gameplay and a Game Over state that resets when "Replay" is pressed.

Task 3

Topic: Building the puzzle mechanic

Scenario

Now it’s time to bring the real gameplay to life.

Task

- Implement the collection behavior as described in the **Gameplay** section.
- The grid must be dynamically generated with random initial values.
- Players interact by clicking/tapping blocks.
- Adjacent blocks of the same color are collected recursively.
- Wait 1s between collection and grid refill.
- Remove the "Make Move" test button from Task 2.

Expected Result

A complete game loop:

- Players can collect blocks and earn points.
- They start with 5 moves, and when moves run out, the Game Over screen appears.
- Restarting the game should restore the same experience.
- The loop can be repeated indefinitely.

Thanks for taking the time to do this test. We know it asks for effort, but we also believe it's a great chance for you to **show off your creativity, technical skills, and problem-solving mindset**.

Good luck - and most importantly, **have fun building it!** 🚀 🎮