Title and metadata

- Math Challenge Mini-Game — Deliverables Doc
- Platform: Unity 2021+ (Unity 6 LTS preferred), Android, New Input System only
- Orientation: Landscape, UI scaled for 16:10–21:9

Flow of code

- Startup: Application loads a single scene with core objects: GameManager, EquationGenerator, QuestionTimer, UIController, EventSystem (Input System UI).
- Menu: MenuPanel is shown; player selects mode (Addition or Subtraction). GameManager initializes Lives = 3, Score = 0, and sets questionTimeSeconds = 15.
- Question loop:
    - Generate a valid equation within bounds 1≤a,b≤201 \le a,b \le 201≤a,b≤20 and ensure result within 1≤answer≤201 \le \text{answer} \le 201≤answer≤20.
    - Show equation on GamePanel, clear answer field, and start countdown (15s) via QuestionTimer.
    - Input arrives via on-screen keypad or UI submit; answer is parsed and sent to GameManager.
    - Correct: Score += 1 → immediately proceed to the next question.
    - Wrong: Lives −= 1 → if Lives > 0, next question; else transition to End.
    - Timeout (no input when timer hits 0): treat as wrong → Lives −= 1 → end or continue as above.
- End screen: EndPanel displays final Score and provides Restart (return to MenuPanel) or Back to Menu.
- Core states: Menu → Playing → End, managed by a simple state machine in GameManager.

Example game loop pseudocode:

```
state = Menu;
onSelectMode(mode):
    state = Playing;
    lives = 3; score = 0;
    nextQuestion();

nextQuestion():
    eq = generator.Generate(mode, 1, 20);
    ui.Show(eq);
    timer.Start(15, onTimeout);

submitAnswer(val):
    timer.Stop();
    if (val == eq.answer) { score++; nextQuestion(); }
    else { loseLife(); }

onTimeout():
    loseLife();

loseLife():
    lives--;
    if (lives <= 0) endGame();
    else nextQuestion();

endGame():
    state = End;
    ui.ShowEnd(score);
```

Tuning (timing, feedback, pacing)

- Timing: Default question time set to 15s to balance pressure and accessibility; supports fast iteration for QA and device variability. Consider exposing 10–20s range for different difficulty presets.

- Feedback:

    - Correct answer: brief color flash (e.g., green), subtle scale “pop” on answer field, optional short tick SFX; total feedback duration 200–300 ms to keep pace.

    - Wrong/timeout: brief red flash and optional vibration/haptic pulse; avoid long lockouts to maintain flow.

- Pacing:

    - Minimal delay between questions (150–300 ms) preserves momentum while making feedback noticeable.

    - Consistent timer visualization (radial or bar) builds urgency; fillAmount updates every frame for smoothness.

    - Optional early-submit: if the entered value cannot be > 20, allow quick submit to reduce friction.

Editor control (exposed variables for quick iteration)

- GameManager

    - startingLives (default 3)

    - questionTimeSeconds (default 15)

    - minValue, maxValue (default 1, 20)

    - allowAddition, allowSubtraction toggles (if future modes are added)

    - nextQuestionDelaySeconds (default 0.2)

- EquationGenerator
    - enforceAnswerBounds (bool)
    - additionRule and subtractionRule strategies (future extensibility)
- QuestionTimer
    - timerImage reference, fillDirection, easing toggle (linear vs. eased UI animation)
- UIController
    - successColor, errorColor, neutralColor
    - successSfx, errorSfx (AudioClips), sfxVolume
    - hapticsEnabled (bool), feedbackDurations (ms)
    - fontSizes and layout scale multipliers for readability across 16:10–21:9
- Build/Platform toggles
    - useMobileHaptics, useLowPerfMode (reduced effects on low-end devices)

ScriptableObjects (configs/themes/difficulty without code changes)

- GameConfigSO
    - Fields: startingLives, minValue, maxValue, questionTimeSeconds, nextQuestionDelaySeconds, allowAddition, allowSubtraction.
    - Purpose: centralize runtime tunables for designers; swap configs to test difficulty quickly.
- ThemeSO
    - Fields: successColor, errorColor, timerColor, backgroundColor, font assets.
    - Purpose: rapid UI reskin or accessibility themes (high contrast).

- AudioConfigSO
    - Fields: successSfx, errorSfx, uiClickSfx, sfxVolume.
    - Purpose: swap audio sets per theme or target market.
- DifficultyCurveSO (optional)
    - Fields: timeByScore curve, livesByScore curve.
    - Purpose: dynamically tighten timer or adjust lives as score increases to scale challenge.

Example ScriptableObject definitions:

```
[CreateAssetMenu(menuName="Config/GameConfig")]
public class GameConfigSO : ScriptableObject {
    public int startingLives = 3;
    public int minValue = 1;
    public int maxValue = 20;
    public float questionTimeSeconds = 15f;
    public float nextQuestionDelaySeconds = 0.2f;
    public bool allowAddition = true;
    public bool allowSubtraction = true;
}

[CreateAssetMenu(menuName="Config/Theme")]
public class ThemeSO : ScriptableObject {
    public Color successColor = Color.green;
    public Color errorColor = Color.red;
    public Color timerColor = Color.white;
    public Color backgroundColor = new Color(0.1f,0.1f,0.1f);
    public TMP_FontAsset font;
}
```

Usage pattern:

- Reference GameConfigSO and ThemeSO on GameManager/UIController via inspector.

- On scene start, apply ThemeSO to UI colors and fonts; read GameConfigSO to initialize lives, timer, and bounds.

- Designers can duplicate and tweak SO assets to create variants (e.g., Easy/Normal/Hard, Dark/Light theme).

Architecture (single-scene layout with references to screenshots)

- Scene organization (as shown in Screenshot_SceneHierarchy.png):
    - GameRoot
        - GameManager (state machine: Menu → Playing → End)
        - EquationGenerator (valid addition/subtraction within bounds)
        - QuestionTimer (controls Image.fillAmount countdown)
        - UI
            - Canvas (Screen Space – Overlay, Canvas Scaler: Scale With Screen Size, reference 1920×1080, Match ~0.5)
            - MenuPanel (mode buttons)
            - GamePanel (equation text, answer field, keypad, timer image, score/lives)
            - EndPanel (final score, restart/menu buttons)
        - EventSystem (Input System UI Input Module)
- Responsibilities:
    - GameManager: orchestrates flow, tracks score/lives, transitions panels, triggers nextQuestion/endGame.
    - EquationGenerator: creates constrained equations ensuring 1≤answer≤201 \le \text{answer} \le 201≤answer≤20 for both addition and subtraction.

    - QuestionTimer: starts/stops 15s countdown; invokes timeout callback on expiry; drives timer UI fill.

    - UIController: shows equation, updates score/lives, collects numeric input, routes submit/clear, and displays end screen.

- Data flow:

    - UIController → GameManager: submitAnswer(), select mode, restart.

    - GameManager → UIController: showEquation(), updateScore(), updateLives(), showEnd().

    - GameManager ↔ QuestionTimer: start/stop countdown, onTimeout callback.

    - GameManager → EquationGenerator: request new equation per mode.