Assembly Project: Columns

Raphael Ramesar
Nehan Punjani

November 26, 2025

1 Instructions and Summary

1. **Which milestones were implemented?**

   We successfully implemented Milestones 1, 2, 3, **4**, and **5**.

   - **Milestone 1:** Basic grid drawing and spawning a random three–high column in the centre of the playing field.
   - **Milestone 2:** Implementing basic movement with W, A, S, D, Q with according movement. Painting screen at 60hz.
   - **Milestone 3:** Detecting and deleting matches of three or more in horizontal, vertical, and diagonal directions. Auto game over when a stack of columns reaches the top.

   For the Final Demonstration requirements:

   - **Milestone 4 (game features):** We implemented more than five easy features, so we satisfy this milestone.
   - **Milestone 5 (more game features):** We implemented at least eight easy features, matching option (a) in the handout.

   We implemented the following:

   - **Easy Feature 1:** Gravity, every tick, the column automatically drops one row after a configurable number of frames.
   - **Easy Feature 2:** Increasing gravity speed, over time, the delay between automatic drops decreases until a minimum speed is reached, this is variable.
   - **Easy Feature 3:** Difficulty selection, before the game begins, the player chooses Easy / Medium / Hard in the console, which sets the initial gravity speed.
   - **Easy Feature 4:** Game Over screen and restart, when a column lands at the top play row, an "OVER / RESTART" screen appears, and pressing `r` restarts a fresh game, keeping no memory of the previous one.
   - **Easy Feature 6:** Pause, pressing `p` displays a blinking "PAUSED" message and freezes the game until `p` is pressed again.
   - **Easy Feature 7:** Outline of dropping column, a three–cell outline shows where the current column will land if dropped.
   - **Easy Feature 10:** Preview panel, a panel on the side shows the next column that will appear.
   - **Easy Feature 11:** Larger preview panel, the same panel displays the next 5 upcoming columns and updates as new columns spawn.

2. **How to run and view the game**

   (a) Open the `columns.asm` file in MARS/Saturn.

   (b) Connect the Bitmap Display with:

- Unit width: 8 pixels
- Unit height: 8 pixels
- Display width: 256 pixels
- Display height: 256 pixels
- Base address: `0x10008000`

(c) Connect the Keyboard and set the base address to `0xffff0000`.

(d) Assemble and run the program with `main` as the entry point.

3. **How to Play**
   - The playing field is a 6 x 13 grid in the left half of the bitmap display. The player controls a falling column of three coloured gems.
   - At all times there is a preview panel on the right that shows the next several columns that will appear.
   - The goal is to make horizontal, vertical, or diagonal lines of at least three gems of the same colour. When a match of length three or more is formed:
     (a) All matched gems are marked in an off–screen copy of the bitmap.
     (b) The matched gems are removed from the main bitmap.
     (c) Any unsupported gems above are dropped down by our gravity routine.
     (d) The process repeats until there are no remaining matches.
   - The game ends when a falling column lands at the top play row; an "OVER / RESTART" screen appears and the player can start a new game.
4. **Controls**
   - `a` — move column left (if not blocked by the wall or other gems)
   - `s` — move column down by one row
   - `d` — move column right
   - `w` — rotate the three gem colours in the column
   - `p` — pause / unpause the game (blinking "PAUSED" text)
   - `q` — quit the program
   - `r` — on the Game Over screen, start a brand new game (re–prompts for difficulty)

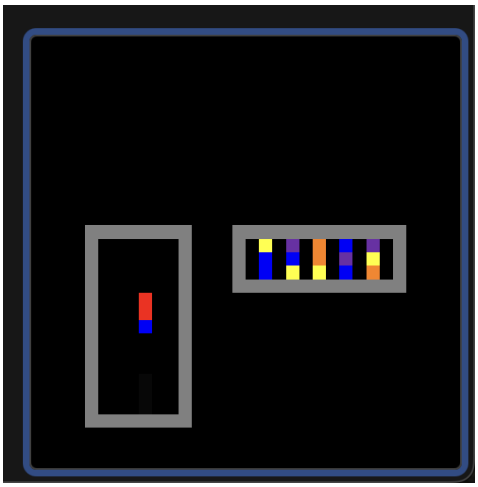

Figure 1: Screenshot of the live Columns game.

2 Attribution Table

Raphael Ramesar, 1011069736	Nehan Punjani, 1010928141
Milestone 1 implementation	Milestone 1 implementation
Milestone 2 implementation	Milestone 2 implementation
M3 – Vertical and horizontal match detection and deletion	M3 – Diagonal down-left and down-right match detection and deletion
M3 – Gravity after match found	M3 – Auto quit when columns reach the top
Easy Features 6, 7, 10, 11 (pause, outline, preview panel, extended preview)	Easy Features 1, 2, 3, 4 (gravity, faster gravity, difficulty select, game over + restart)
Milestone 4 and 5 feature integration and testing	Milestone 4 and 5 feature integration and testing
Testing and debugging	Testing and debugging
Documentation and demo preparation	Documentation and demo preparation

3 Implementation Details

This section summarizes some of the core algorithms and features using the design sketches, pixel layouts, and development screenshots we produced during the project. Each image highlights a key part of the logic or rendering pipeline.

3.1 Horizontal Match Detection

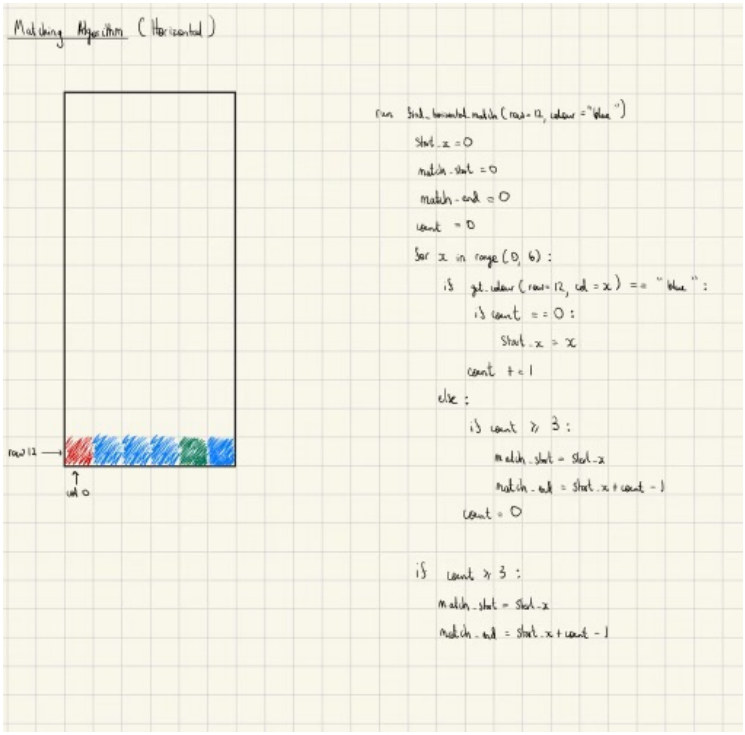

Figure 2: Sketch of the horizontal matching algorithm: scanning a row, identifying runs of 3+ same-coloured blocks, and marking their positions.

3.2 Pause Screen Pixel Layout

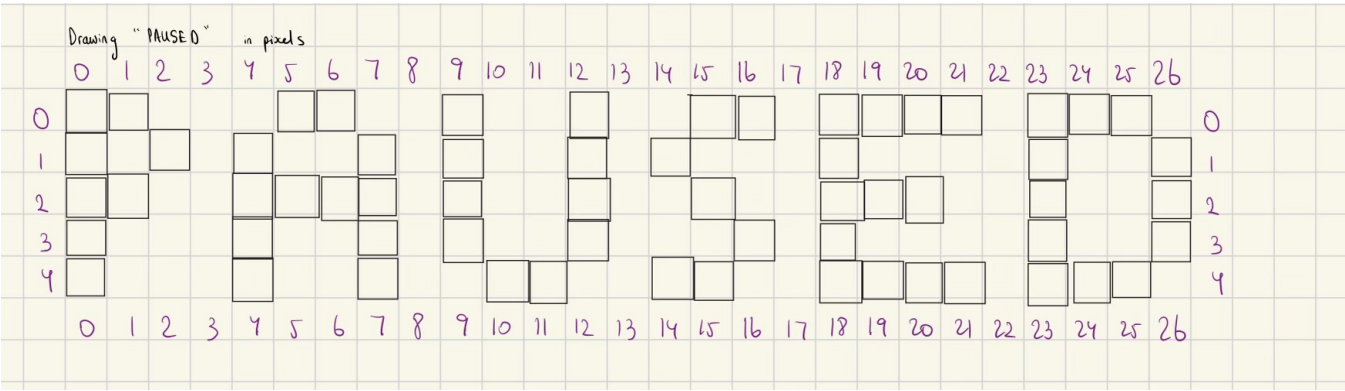

Figure 3: Pixel-by-pixel layout for drawing “PAUSED” in the bitmap display.

3.3 Dynamic Gravity Speed-Up

```
update_gravity_speed:

    # Increase "time since last speed-up"
    lw  $t0, gravity_level_timer
    addi $t0, $t0, 1
    sw  $t0, gravity_level_timer

    # Check if it's time to speed up (e.g., every 600 frames)
    # 600 * 17 ms ≈ 10 seconds of real time
    li  $t1, 600
    blt $t0, $t1, ugs_return        # if timer < 600, nothing to do yet

    # Time to speed up gravity
    sw  $zero, gravity_level_timer  # reset timer

    # Load current speed and min speed
    lw  $t2, gravity_speed
    lw  $t3, gravity_min_speed

    # Decrease gravity_speed by 5 (faster), but don't go below min
    addi $t2, $t2, -10
    bge  $t2, $t3, ugs_store        # if new speed >= min, keep it
    add  $t2, $zero, $t3            # else clamp to min
```

Figure 4: Gravity update routine: every 600 ticks, gravity_speed is reduced (faster falling) until reaching a minimum cap.

3.4 Difficulty Selection

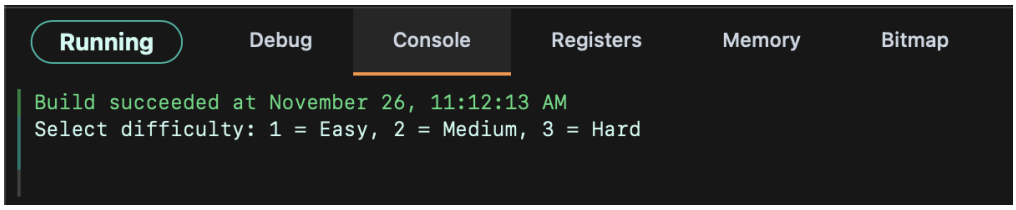

Figure 5: Difficulty chooser in the console: Easy, Medium, and Hard set different initial gravity timings.

3.5 Game Over Screen

Figure 6: Pixel-drawn OVER / RESTART screen with colour accents.

3.6 Preview Panel (In-Game)

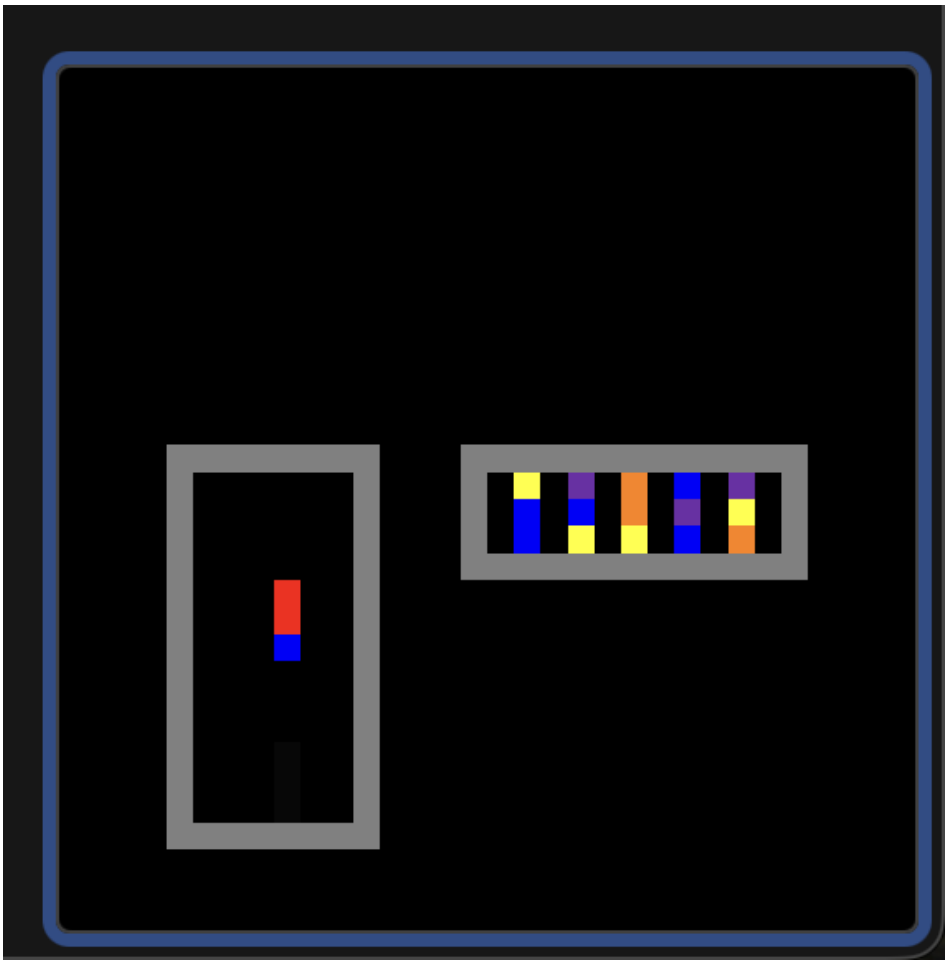

Figure 7: Preview panel showing the next 4–5 columns. Updated whenever a new column spawns.

3.7 Preview Panel Memory Layout

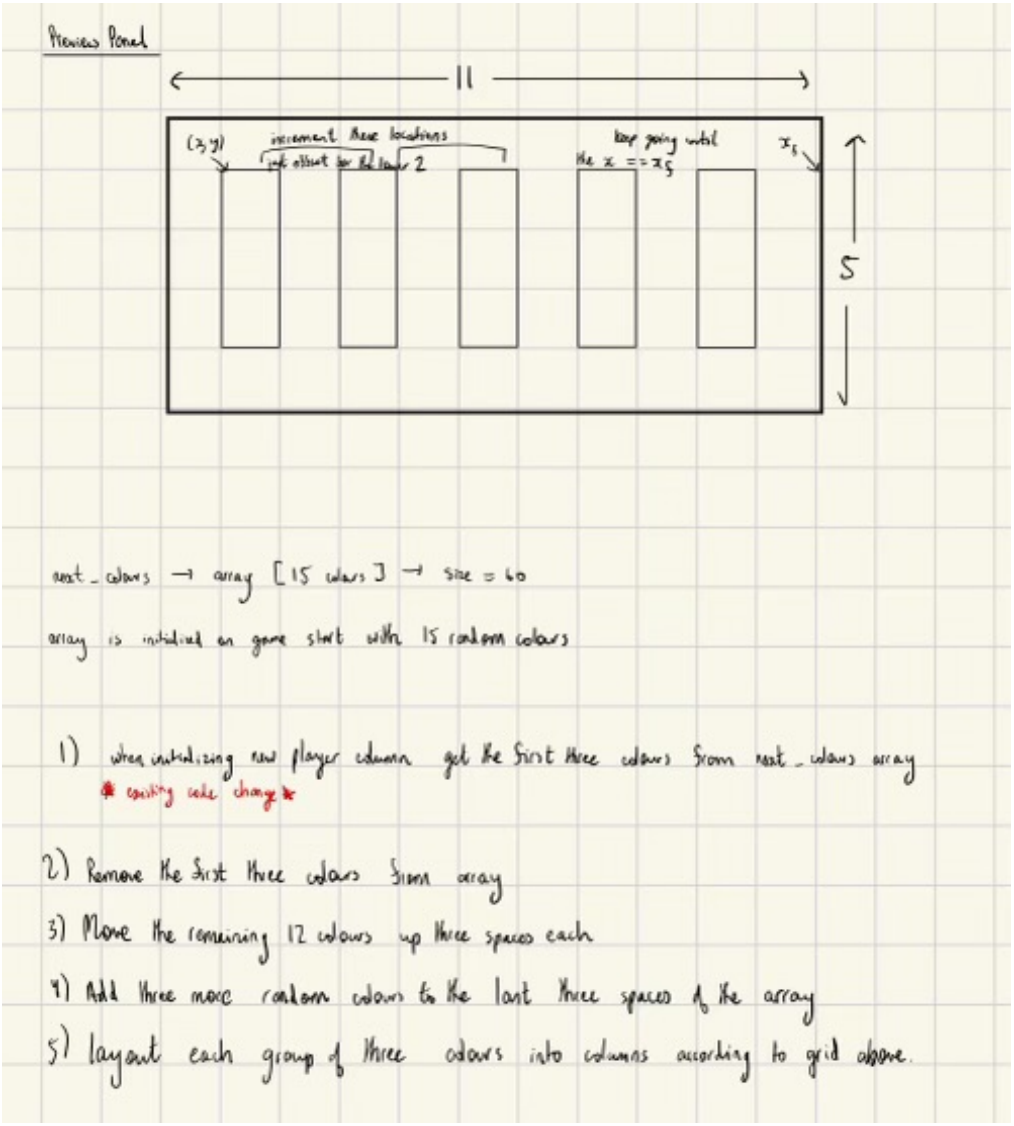

Figure 8: Design plan for preview-panel memory structure: an array of 15 colours grouped into five 3-block column previews.

3.8 Pause Rendering (Final Output)

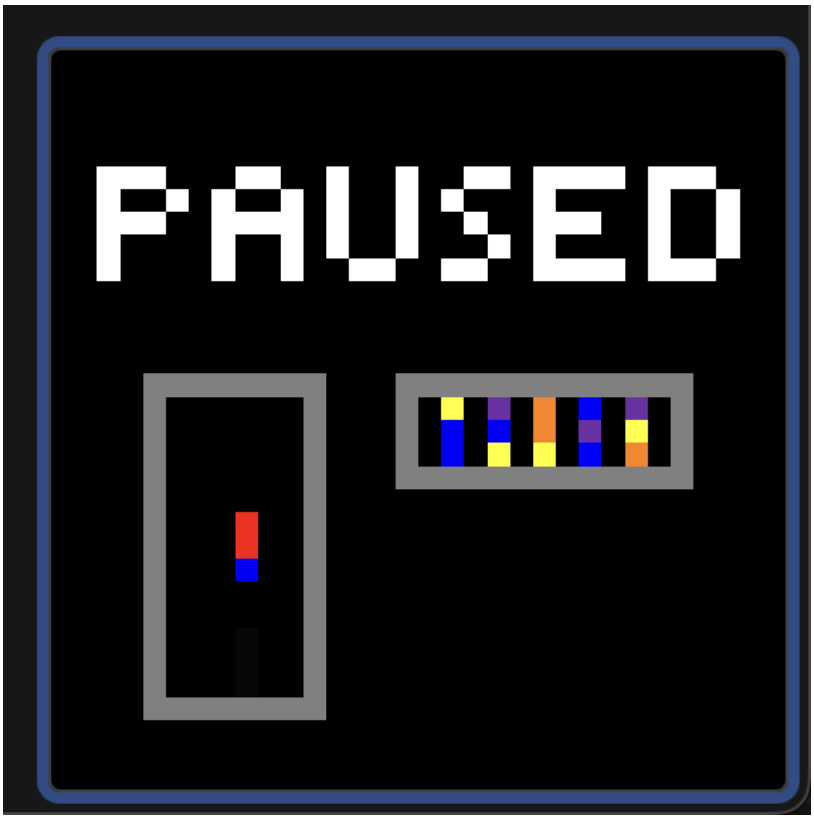

Figure 9: Final "PAUSED" rendering with blinking effect in-game.

3.9 Gravity Variables

```
gravity_counter: .word 0          # counts ticks until next automatic drop
gravity_speed:        .word 60     # current number of ticks between drops
gravity_min_speed:    .word 15     # fastest allowed speed (smaller = faster)
gravity_level_timer:  .word 0      # counts ticks until we speed up gravity

# Gravity difficulty settings (ticks between automatic drops)
EASY_TICKS:      .word 80     # slowest
MEDIUM_TICKS:    .word 60     # normal
HARD_TICKS:      .word 40     # fastest
```

Figure 10: Gravity-related variables: current speed, minimum speed, counter, and difficulty presets.