aditijb@stanford.edu, anjalisr@stanford.edu
Win 2024 | CS107E Final Project

Tetris

Parts/Physical modules

Handheld controller device (think Wii remote but simpler) Aditi's HW notes

- MangoPi (microcontroller) **(processor)**
- Accelerometer (tilt) **(input)**
- Buttons (as necessary) **(input)**
- Servo (for vibration/haptic feedback) **(output)**
- Connects to monitor via. long hdmi cable (tethered controller)

Monitor

- Displays game element **(output)**

Functionality

Elevator pitch:

- Tetris but with a handheld controller

Level 0: (separate)

- Get accelerometer module working (Julie's + wrapper module)
- Controller breadboarded
- Can set-up and draw a simple game on monitor
  - Blocks
  - Animate dropping blocks

Level 1: (basic integration; working product)

- Use interrupts to store pending moves from controller
- Moving the blocks by
  - Tilt right => move cur block to the right by 1 column
  - Tilt left => move cur block to the right by 1 column
  - Tilt down/back up quickly => move block all the way down and lock in place immediately
- Graphically handles clear-line for full lines, overflowed grid, and other usual cases of the game
- Speeds up how quickly
- Generally, a working system. Nice looking display; decent re-draw speeds. Reasonable measurement of sensor inputs.

Level 2: (fun stuff)

- Servo (haptic feedback) for invalid/illegal moves (works like shell_bell())
- Casing for the "remote" module
- Can disconnect mango pi from laptop after downloading and the program still runs
- "Replay" screen
- Animations for row clears

Level 3: (would be nice)

- Dark mode
- Menu (display & access via. buttons)
- Shake remote to swap cur block with the 1st in queue (more sensors!)

- Bluetooth (get rid of the hdmi tether to the controller!)
- Theme song (We need to order a piezo (sound output) but Daniel suggested we play the Tetris theme too https://dragaosemchama.com/en/2019/02/songs-for-arduino/)

Anjali - Graphics Software Functionality Architecture + Feature Brainstorm

- 2D array sized # rows x # cols of tetris grid – store which piece is where
    - Redraw only what changes each time and then swap buffer
    - Clear function that takes in coordinate array of which coordinates to clear
- Queue for pieces to be enqueued (random selection)
- up arrow --> clockwise rotation of falling piece by 90 degrees
    - handle with interrupts (falling motion is interrupted)
    - rotation algorithm – calculate new coordinate position of each block piece using mat-vec multiplication (we know clockwise rotation by 90 degree matrix!)
        - § potentially cache – save all transformations in constant array/struct to save time?
- Right, left, down arrow – motion by increments of single grid square
    - When piece falls, once it hits bottom level, user should still be able to shift L/R once if they want; then only piece position gets locked (use timer to add extra time buffer?)
- Single + double row clearing; pieces above "drop"
    - Writing bottom to top in 2D array
    - Keep track of "head row" in 2D array
    - Memcopy and shift everything above cleared row(s) down by num rows cleared
    - clear row(s) starting at head row – number determined by num rows cleared
    - Vibration for this????
- Scoring system ???
    - Need to look into the way tetris does it, or we can come up with our own?
- Lower priority / if time permits
    - Aesthetic – gradient fill squares :)
    - Some sort of power up?

<u>GAME COMPLEXITY LAYERS</u>

Layer 1 / P1 –

- Clean falling of colored tetris pieces into spot
- Smooth left/right/down motion (accelerometer)
- Single row clearing – clear bottom most filled row
- 3D printed remote control
- Scoring

Layer 2 / P2 –

- Ability to rotate (increments of 90 deg clockwise) falling tetris piece (button)
- Single AND double row clearing support – vibration from servo

- Visible queue of next piece
- Gradient/outline designs on tetris pieces :)

Layer 3 / P3 –

- Some creative power up?
- Color mode switching?