# Breakout tutorial (intermediate)

## Files

In the Games folder:

- Create a new folder called `Breakout`.

- In the same folder, add the files:

  - `Breakout.ino` (main loop)
  - `Pins.h` (all pins, sizes, constants)
  - `Input.h/.cpp` (debounce + joystick repeat + “serve pressed”)
  - `Game.h/.cpp` (all state + physics + brick logic)
  - `Render.h/.cpp` (all drawing + score digits)

---

## Pins.h pseudocode

- Define all pins and fixed numbers:

  - LED data pin, button pins, joystick pins
  - grid dimensions 10x20
  - paddle width, brick zone rows
  - time step constants (frame ms, ball ms, brick drop ms)
  - joystick repeat constants

---

## Input pseudocode

### Data

- A `Btn` that tracks: stable pressed state, previous stable state, debounce timer
- Global instances for 4 buttons and 4 joystick directions
- Repeat timers/state for left and right

### Functions

- `Input_begin()`:

  - set all pins to `INPUT_PULLUP`
  - reset repeat timers

- `Input_update()`:

  - debounce each input once per frame

- `Input_servePressedEdge()`:

  - return true if ANY serve button had a press edge this frame

- `Input_paddleStepFromJoystickRepeat(now)`:
  - if left held: return -1 on initial press, then -1 again at repeat intervals
  - if right held: return +1 similarly
  - else return 0
- `Input_latch()`:
  - copy stable -> previous stable so edges work next frame

---

## Game pseudocode

### State

- `bricksGrid[y][x]` colour or empty
- paddle position `paddleX`
- ball position `ballX, ballY`
- ball velocity `ballVX, ballVY`
- `ballStuck` boolean
- timers: `tBall`, `tBrickDrop`
- score + speed-up counters
- `wheelPos` for row colours

### Reset

- clear score and gameOver
- fill bricks rows 0..7 with coloured rows
- set ball on paddle center, stuck = true
- reset timers and speeds

### Paddle move

- add dx then clamp to [0..W-PADDLE_W]
- if ball stuck, move ball with paddle

### Serve

- if stuck, unstuck and start ball timer

### Brick drop tick (every 15s)

- if bricks already on bottom brick row -> game over
- shift rows down by one
- create new row at top with wheel colour (uniform per row)

### Ball step

- compute next position from velocity

- bounce off left/right edges

- bounce off top edge
- if next cell contains brick:
  - remove brick
  - score += 10
  - maybe speed up ball
  - invert vertical direction (simple)
- check paddle collision:
  - if ball enters paddle row and overlaps paddle columns, bounce upward
  - adjust horizontal direction based on hit position
- if ball goes below paddle row -> game over
- commit position

---

## Render pseudocode

- clear background
- draw bricks from bricksGrid
- draw paddle
- draw ball (unless game over)
- if game over: fill screen red
- update score digits when score changes (called by game)
- push pixels: `lcdPanel->render()`, `pixelGrid->render()`, `strip.show()`

---

## .ino pseudocode

- `setup`:
  - init serial, seed RNG
  - init renderer (strip, pixelGrid, lcdPanel)
  - init input
  - `Game_reset()`
- `loop`:
  - enforce fixed frame time (FRAME_MS)
  - `Input_update()`
  - if button press edge:
    - if game over -> reset
    - else -> serve
  - if brick drop timer expired -> `Game_brickDropTick()`
  - step = joystick repeat; if step != 0 -> `Game_movePaddle(step)`

    - if ball moving and timer expired -> `Game_stepBallOnce()`
    - render frame
    - latch inputs