```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <meta name="viewport"
content="width=device-width,initial-scale=1,maximum-scale=1,user-scalable=no" />
  <title>Flappy</title>
  <style>
    html,body{margin:0;height:100%;background:#111;font-family:system-ui,Segoe
UI,Roboto,Arial;}
    canvas{display:block;margin:0 auto;background:linear-gradient(#70c5ce,#b6e6ff
60%,#d2f5c8 60%,#d2f5c8);touch-action:manipulation;}

.hud{position:fixed;left:0;right:0;top:10px;text-align:center;color:#fff;font-weight:700;text-shadow:
0 2px 6px rgba(0,0,0,.5);pointer-events:none;}

.hint{position:fixed;left:0;right:0;bottom:18px;text-align:center;color:#fff;opacity:.9;text-shadow:0
2px 6px rgba(0,0,0,.5);pointer-events:none;}

.btnrow{position:fixed;left:0;right:0;bottom:60px;display:flex;justify-content:center;gap:10px;point
er-events:auto;}
    button{border:0;border-radius:12px;padding:10px 14px;font-weight:700;cursor:pointer}
  </style>
</head>
<body>
  <div class="hud" id="hud">Score: 0</div>
  <div class="btnrow">
    <button id="restart" style="display:none;">Restart</button>
  </div>
  <div class="hint" id="hint">Tap / Space to flap • R to restart</div>
  <canvas id="c"></canvas>

<script>
(() => {
  const canvas = document.getElementById("c");
  const ctx = canvas.getContext("2d");
  const hud = document.getElementById("hud");
  const hint = document.getElementById("hint");
  const restartBtn = document.getElementById("restart");

  // Responsive canvas (fixed aspect-ish)
  function resize() {
    const w = Math.min(window.innerWidth, 520);
    const h = Math.min(window.innerHeight, 820);
  canvas.width = w;
  canvas.height = h;
}
window.addEventListener("resize", resize);
resize();

// Game constants (scaled by canvas size)
const G = () => canvas.height * 0.0021;          // gravity
const FLAP = () => canvas.height * 0.035;        // flap impulse
const SPEED = () => canvas.width * 0.0043;       // pipe speed
const GAP = () => canvas.height * 0.22;          // gap size
const PIPE_W = () => canvas.width * 0.16;

// State
let bird, pipes, score, best, started, dead, lastTime;

// Load best score
best = Number(localStorage.getItem("flappy_best") || 0);

function reset() {
  bird = {
    x: canvas.width * 0.28,
    y: canvas.height * 0.45,
    r: Math.max(10, canvas.width * 0.03),
    vy: 0,
  };
  pipes = [];
  score = 0;
  started = false;
  dead = false;
  lastTime = performance.now();
  hud.textContent = `Score: ${score} • Best: ${best}`;
  hint.textContent = "Tap / Space to flap • R to restart";
  restartBtn.style.display = "none";
  spawnPipe(true);
}

function rand(min, max){ return Math.random() * (max - min) + min; }

function spawnPipe(initial=false) {
  const margin = canvas.height * 0.14;
  const center = rand(margin + GAP()/2, canvas.height - margin - GAP()/2);
  const x = initial ? canvas.width * 0.9 : canvas.width + PIPE_W();
  pipes.push({
    x,
    gapY: center,
    scored: false
  });
}

function flap() {
  if (dead) return;
  if (!started) started = true;
  bird.vy = -FLAP();
}

function die() {
  if (dead) return;
  dead = true;
  best = Math.max(best, score);
  localStorage.setItem("flappy_best", String(best));
  hud.textContent = `Score: ${score} • Best: ${best}`;
  hint.textContent = "Game over. Press R or Restart.";
  restartBtn.style.display = "inline-block";
}

function collideCircleRect(cx, cy, r, rx, ry, rw, rh) {
  const nx = Math.max(rx, Math.min(cx, rx + rw));
  const ny = Math.max(ry, Math.min(cy, ry + rh));
  const dx = cx - nx, dy = cy - ny;
  return dx*dx + dy*dy <= r*r;
}

function update(dt) {
  // idle bob before start
  if (!started && !dead) {
    bird.y = canvas.height * 0.45 + Math.sin(performance.now()/250) * (canvas.height*0.008);
    bird.vy = 0;
    return;
  }

  // physics
  bird.vy += G() * dt;
  bird.y += bird.vy * dt;

  // ground / ceiling
  const groundY = canvas.height * 0.92;
  if (bird.y + bird.r > groundY) { bird.y = groundY - bird.r; die(); }
  if (bird.y - bird.r < 0) { bird.y = bird.r; bird.vy = 0; }

  // pipes move
  const v = SPEED() * dt;
  for (const p of pipes) p.x -= v;

  // spawn new pipes
  const last = pipes[pipes.length - 1];
  if (last && last.x < canvas.width * 0.55) spawnPipe();

  // remove offscreen
  while (pipes.length && pipes[0].x + PIPE_W() < -10) pipes.shift();

  // scoring + collision
  for (const p of pipes) {
    const topH = p.gapY - GAP()/2;
    const botY = p.gapY + GAP()/2;
    const w = PIPE_W();

    // score when bird passes center of pipe
    if (!p.scored && p.x + w < bird.x) {
      p.scored = true;
      score++;
      hud.textContent = `Score: ${score} • Best: ${best}`;
    }

    // collision with top pipe
    if (collideCircleRect(bird.x, bird.y, bird.r, p.x, 0, w, topH)) die();
    // collision with bottom pipe
    if (collideCircleRect(bird.x, bird.y, bird.r, p.x, botY, w, canvas.height - botY)) die();
  }
}

function draw() {
  ctx.clearRect(0,0,canvas.width,canvas.height);

  // ground
  const groundY = canvas.height * 0.92;
  ctx.fillStyle = "rgba(90,200,90,0.95)";
  ctx.fillRect(0, groundY, canvas.width, canvas.height-groundY);
  ctx.fillStyle = "rgba(60,160,60,0.95)";
  ctx.fillRect(0, groundY-10, canvas.width, 10);

  // pipes
```

```
for (const p of pipes) {
  const w = PIPE_W();
  const topH = p.gapY - GAP()/2;
  const botY = p.gapY + GAP()/2;

  // pipe body
  ctx.fillStyle = "#2ecc71";
  ctx.fillRect(p.x, 0, w, topH);
  ctx.fillRect(p.x, botY, w, canvas.height - botY);

  // pipe lips
  ctx.fillStyle = "#27ae60";
  ctx.fillRect(p.x - 4, topH - 16, w + 8, 16);
  ctx.fillRect(p.x - 4, botY, w + 8, 16);
}

// bird
ctx.save();
const tilt = Math.max(-0.8, Math.min(0.8, bird.vy / (canvas.height*0.03)));
ctx.translate(bird.x, bird.y);
ctx.rotate(tilt);

ctx.fillStyle = "#f1c40f";
ctx.beginPath();
ctx.arc(0,0,bird.r,0,Math.PI*2);
ctx.fill();

// beak
ctx.fillStyle = "#e67e22";
ctx.fillRect(bird.r*0.6, -bird.r*0.18, bird.r*0.9, bird.r*0.36);

// eye
ctx.fillStyle = "#fff";
ctx.beginPath();
ctx.arc(bird.r*0.15, -bird.r*0.35, bird.r*0.28, 0, Math.PI*2);
ctx.fill();
ctx.fillStyle = "#111";
ctx.beginPath();
ctx.arc(bird.r*0.2, -bird.r*0.35, bird.r*0.12, 0, Math.PI*2);
ctx.fill();

ctx.restore();

// start text
```

```
    if (!started && !dead) {
      ctx.fillStyle = "rgba(0,0,0,0.35)";
      ctx.fillRect(0, canvas.height*0.18, canvas.width, canvas.height*0.12);
      ctx.fillStyle = "#fff";
      ctx.font = `700 ${Math.max(18, canvas.width*0.055)}px system-ui`;
      ctx.textAlign = "center";
      ctx.fillText("TAP / SPACE TO START", canvas.width/2, canvas.height*0.25);
    }
  }

  function loop(t) {
    const dt = Math.min(2.5, (t - lastTime) / 16.6667); // normalize ~60fps
    lastTime = t;

    if (!dead) update(dt);
    draw();

    requestAnimationFrame(loop);
  }

  // controls
  window.addEventListener("keydown", (e) => {
    if (e.code === "Space") { e.preventDefault(); flap(); }
    if (e.code === "KeyR") reset();
  });
  window.addEventListener("pointerdown", () => flap(), {passive:true});
  restartBtn.addEventListener("click", reset);

  reset();
  requestAnimationFrame(loop);
})();
</script>
</body>
</html>
```