```
<!doctype html>
<html lang="en">
<head>
<meta charset="utf-8"/>
<meta name="viewport" content="width=device-width,initial-scale=1,viewport-fit=cover"/>
<title>Rematch/Retro Goal — Mobile + Desktop (Demo)</title>
<style>
  :root{
    --bg1:#071028; --bg2:#082233; --accent:#00d1ff; --gold:#ffd166;
    --panel:rgba(255,255,255,0.04); --muted:rgba(255,255,255,0.65);
    --btn-gradient:linear-gradient(180deg,#00b8ff,#0077c2);
  }
  html,body{height:100%;margin:0;font-family:Inter,system-ui,-apple-system,Segoe
UI,Roboto,Arial;}

body{background:linear-gradient(180deg,var(--bg1),var(--bg2));color:#fff;display:flex;flex-directio
n:column;align-items:center;padding:12px;gap:10px}
  .wrap{width:100%;max-width:980px}
  header{display:flex;justify-content:space-between;align-items:center}
  h1{font-size:18px;margin:0;color:var(--accent)}
  .sub{font-size:12px;color:var(--muted)}
  #game{background:
linear-gradient(180deg,#05202a,#0b262e);border-radius:14px;padding:10px;box-shadow:0
10px 30px rgba(0,0,0,.6)}

#field{background:linear-gradient(180deg,#0b6b3e,#0b5a36);border-radius:10px;overflow:hidde
n;touch-action:none}
  canvas{display:block;width:100%;height:auto;border-radius:8px}
  .hud{display:flex;justify-content:space-between;align-items:center;padding:8px
6px;font-size:13px;gap:10px}

.panel{background:var(--panel);padding:8px;border-radius:10px;display:flex;gap:8px;align-items
:center}
  .score{font-weight:800;color:var(--gold);font-size:18px}
  .mini{font-size:12px;color:var(--muted)}
  .controls{display:flex;gap:8px;justify-content:center;padding:10px;flex-wrap:wrap}
  .btn{padding:12px
16px;border-radius:12px;border:none;background:var(--btn-gradient);color:#fff;font-weight:700;b
ox-shadow:0 6px 20px rgba(0,120,200,.14)}
  .btn.secondary{background:rgba(255,255,255,0.06)}

.meter{height:8px;background:rgba(255,255,255,0.06);border-radius:6px;overflow:hidden;width:
120px}
  .meter > i{display:block;height:100%;background:linear-gradient(90deg,#ffd166,#ff6b6b)}
```

```
  .footer{font-size:12px;color:var(--muted);text-align:center;padding:6px}
  .kbdHelp{font-size:12px;color:var(--muted);margin-top:6px}
  @media(min-width:900px){body{padding:28px}}
</style>
</head>
<body>
<div class="wrap">
  <header>
    <div>
      <h1>Arcade Soccer — Mobile + Desktop Demo</h1>
      <div class="sub">Touch controls + keyboard (WASD / arrows). Short 60s matches.</div>
    </div>
    <div class="panel">
      <div style="text-align:right">
        <div class="mini">Time</div>
        <div class="score" id="time">60</div>
      </div>
      <div style="width:14px"></div>
      <div style="text-align:right">
        <div class="mini">Score</div>
        <div class="score"><span id="scoreYou">0</span> — <span
id="scoreOpp">0</span></div>
      </div>
    </div>
  </header>

  <div id="game">
    <div id="field" style="width:100%;max-width:900px">
      <canvas id="c" width="720" height="940"></canvas>
    </div>

    <div
style="padding:10px;display:flex;justify-content:space-between;align-items:center;gap:8px;flex-
wrap:wrap">
      <div class="panel">
        <div style="margin-right:8px">
          <div class="mini">Shot Power</div>
          <div class="meter"><i id="powerBar" style="width:30%"></i></div>
        </div>
        <div style="margin-left:6px">
          <div class="mini">Mode</div>
          <div id="mode" style="font-weight:700;color:var(--accent)">Control</div>
        </div>
      </div>
```

```
      <div style="display:flex;gap:8px">
        <button class="btn secondary" id="restart">Restart</button>
        <button class="btn" id="auto">Auto AI</button>
      </div>
    </div>

    <div class="controls" style="justify-content:space-between">
      <div style="display:flex;gap:8px">
        <button class="btn secondary" id="passBtn">PASS (P)</button>
        <button class="btn" id="shootBtn">SHOOT (Space)</button>
      </div>
      <div style="text-align:right" class="kbdHelp">
        <div>Keyboard: WASD / ↑↓←→ to move — R restart</div>
      </div>
    </div>
  </div>

  <div class="footer">Save as <code>index.html</code>. Upload to GitHub Pages or Netlify for a link.</div>
</div>

<script>
/* Arcade Soccer — Mobile + Desktop demo
   - Touch: drag to dribble, tap to pass, swipe to shoot
   - Desktop: WASD/arrows to move, P to pass, Space to shoot
   - 60s match, simple AI, single file
*/

(() => {
  // Canvas & world setup
  const canvas = document.getElementById('c');
  const ctx = canvas.getContext('2d');
  const W = canvas.width, H = canvas.height;
  const PW = 100, PH = 140; // virtual pitch coordinates

  function vtx(x){ return (x / PW) * (W - 40) + 20; }
  function vty(y){ return (y / PH) * (H - 40) + 20; }
  function screenToWorldXY(sx, sy){
    return {
      x: ((sx - 20) / (W - 40)) * PW,
      y: ((sy - 20) / (H - 40)) * PH
    };
  }
```

```
  // State
  const state = {
    player: {x:50, y:110, vx:0, vy:0, speed:1.9, radius:3.8, hasBall:true},
    opponent: {x:50, y:30, vx:0, vy:0, speed:1.6, radius:3.8, hasBall:false},
    teammates: [], // optional for 3v3 later
    ball: {x:50, y:110, vx:0, vy:0},
    scoreYou:0, scoreOpp:0,
    timeLeft:60,
    running:true, autoAI:false,
    power:30,
    mode:'control',
    lastUpdate: performance.now()
  };

  // UI refs
  const timeEl = document.getElementById('time');
  const scoreYouEl = document.getElementById('scoreYou');
  const scoreOppEl = document.getElementById('scoreOpp');
  const powerBar = document.getElementById('powerBar');
  const modeEl = document.getElementById('mode');
  const restartBtn = document.getElementById('restart');
  const autoBtn = document.getElementById('auto');
  const passBtn = document.getElementById('passBtn');
  const shootBtn = document.getElementById('shootBtn');

  // Utilities
  function clamp(v,a,b){ return Math.max(a,Math.min(b,v)); }
  function dist(a,b){ return Math.hypot(a.x-b.x, a.y-b.y); }

  // Reset positions
  function resetRound() {
    state.player.x = 50; state.player.y = PH - 20; state.player.vx = 0; state.player.vy = 0;
state.player.hasBall = true;
    state.opponent.x = 50; state.opponent.y = 30; state.opponent.vx = 0; state.opponent.vy = 0;
state.opponent.hasBall = false;
    state.ball.x = state.player.x; state.ball.y = state.player.y; state.ball.vx = 0; state.ball.vy = 0;
  }
  resetRound();

  // Input handling (pointer for mobile + mouse)
  let pointer = {down:false, x:0, y:0, startX:0, startY:0, startT:0, dragging:false};

  canvas.addEventListener('pointerdown', e => {
```

```
  const rect = canvas.getBoundingClientRect();
  pointer.down = true;
  pointer.x = e.clientX - rect.left; pointer.y = e.clientY - rect.top;
  pointer.startX = pointer.x; pointer.startY = pointer.y; pointer.startT = Date.now();
  // if pointer near player and player has ball -> start dribble
  const w = screenToWorldXY(pointer.x, pointer.y);
  if(dist(w, state.player) < 8 && state.player.hasBall){
    pointer.dragging = true;
    state.mode = 'dribble';
    modeEl.textContent = 'Dribble';
  }
});

canvas.addEventListener('pointermove', e => {
  if(!pointer.down) return;
  const rect = canvas.getBoundingClientRect();
  pointer.x = e.clientX - rect.left; pointer.y = e.clientY - rect.top;
});

canvas.addEventListener('pointerup', e => {
  pointer.down = false;
  const rect = canvas.getBoundingClientRect();
  const upx = e.clientX - rect.left, upy = e.clientY - rect.top;
  const dx = upx - pointer.startX, dy = upy - pointer.startY;
  const dt = Date.now() - pointer.startT;
  const swipeDist = Math.hypot(dx,dy);
  const worldUp = screenToWorldXY(upx, upy);

  if(swipeDist > 30 && dt < 500 && state.player.hasBall){
    // shoot toward swipe direction: swipe angle
    const ang = Math.atan2(dy, dx);
    shootBall(ang, clamp(state.power + swipeDist/3, 10, 140)/25);
  } else {
    if(state.player.hasBall){
      // tap to pass/kick toward clicked point
      kickBallTo(worldUp.x, worldUp.y, 0.9 + state.power/240);
    } else {
      // if not in possession, move player toward clicked point
      // convert world coords to player velocity for a short dash
      const tx = worldUp.x, ty = worldUp.y;
      const dxw = tx - state.player.x, dyw = ty - state.player.y;
      state.player.vx += dxw * 0.03;
      state.player.vy += dyw * 0.03;
    }
```

```
  }
  pointer.dragging = false;
  state.mode = 'control';
  modeEl.textContent = 'Control';
});

// Keyboard controls
const keys = {};
window.addEventListener('keydown', e => {
  keys[e.key.toLowerCase()] = true;
  // quick action keys
  if(e.key === ' '){ // space -> shoot
    e.preventDefault();
    if(state.player.hasBall) {
      const ang = Math.atan2(- (state.player.y), (50 - state.player.x)); // toward top
      shootBall(ang, 1.3 + state.power/80);
    }
  } else if(e.key.toLowerCase() === 'p'){
    if(state.player.hasBall){
      // pass to forward location
      const tx = clamp(state.player.x + (Math.random()-0.5)*14, 8, PW-8);
      const ty = state.player.y - 16;
      kickBallTo(tx, ty, 0.95 + state.power/220);
    }
  } else if(e.key.toLowerCase() === 'r'){
    fullReset();
  }
});
window.addEventListener('keyup', e => { keys[e.key.toLowerCase()] = false; });

// UI buttons
passBtn.addEventListener('click', ()=>{
  if(state.player.hasBall){
    const tx = clamp(state.player.x + (Math.random()-0.5)*14, 8, PW-8);
    const ty = state.player.y - 16;
    kickBallTo(tx, ty, 0.95 + state.power/220);
  }
});
shootBtn.addEventListener('click', ()=>{
  if(state.player.hasBall){
    const ang = Math.atan2(- (state.player.y), (50 - state.player.x));
    shootBall(ang, 1.3 + state.power/80);
  }
});
```

```
restartBtn.addEventListener('click', fullReset);
autoBtn.addEventListener('click', ()=>{
  state.autoAI = !state.autoAI; autoBtn.textContent = state.autoAI ? 'Auto AI: ON' : 'Auto AI';
});

// Ball & action functions
function kickBallTo(tx, ty, force){
  const dx = tx - state.ball.x, dy = ty - state.ball.y;
  const d = Math.hypot(dx,dy) || 1;
  state.ball.vx = (dx/d) * force;
  state.ball.vy = (dy/d) * force;
  state.player.hasBall = false;
  state.opponent.hasBall = false;
}
function shootBall(angle, power){
  state.ball.vx = Math.cos(angle) * power * 1.8;
  state.ball.vy = Math.sin(angle) * power * 1.8;
  state.player.hasBall = false;
  state.opponent.hasBall = false;
}

// AI
function updateAI(dt){
  const opp = state.opponent;
  if(state.autoAI){
    // chase & shoot if opportunity
    if(!state.opponent.hasBall){
      const dx = state.ball.x - opp.x, dy = state.ball.y - opp.y, d = Math.hypot(dx,dy)||1;
      opp.vx += (dx/d) * 0.03;
      opp.vy += (dy/d) * 0.03;
    } else {
      // if near bottom, shoot toward bottom goal
      if(opp.y > PH - 30){
        kickBallTo(50, PH - 2, 1.6);
        state.opponent.hasBall = false;
      } else {
        opp.vy += 0.02; // advance forward
      }
    }
    // occasionally attempt to take ball if near player
    if(!state.opponent.hasBall && dist(state.opponent, state.player) < 5 && Math.random() <
0.12){
      state.opponent.hasBall = true; state.player.hasBall = false;
```

```
      state.ball.x = state.opponent.x; state.ball.y = state.opponent.y; state.ball.vx=0;
state.ball.vy=0;
    }
   } else {
    // conservative AI: mark player and intercept
    const target = {x: state.player.x, y: Math.min(state.player.y, 40)};
    opp.vx += (target.x - opp.x) * 0.02;
    opp.vy += (target.y - opp.y) * 0.02;
    if(!state.opponent.hasBall && dist(state.opponent, state.ball) < 4){
     state.opponent.hasBall = true; state.player.hasBall = false;
     state.ball.x = state.opponent.x; state.ball.y = state.opponent.y; state.ball.vx=0;
state.ball.vy=0;
    }
    if(state.opponent.hasBall && opp.y < 30 && Math.random() < 0.02){
     kickBallTo(50, PH - 2, 1.4); state.opponent.hasBall = false;
    }
   }
  }

  // Physics/integration
  function integrate(obj, dt, speed){
   obj.x += obj.vx * dt;
   obj.y += obj.vy * dt;
   const vmax = 0.95 * speed;
   const v = Math.hypot(obj.vx, obj.vy);
   if(v > vmax){
    obj.vx = (obj.vx/v) * vmax; obj.vy = (obj.vy/v) * vmax;
   }
   // friction
   obj.vx *= 0.85; obj.vy *= 0.85;
  }
  function clampEntity(e){
   e.x = clamp(e.x, 6, PW - 6);
   e.y = clamp(e.y, 6, PH - 6);
  }
  function clampBall(){
   state.ball.x = clamp(state.ball.x, 2, PW - 2);
   state.ball.y = clamp(state.ball.y, 2, PH - 2);
  }

  function respawnAfterGoal(playerScored){
   state.running = false;
   setTimeout(()=>{
    if(playerScored){
```

```
      state.player.x = 50; state.player.y = PH - 20; state.player.hasBall = false;
      state.opponent.x = 50; state.opponent.y = 30; state.opponent.hasBall = true;
      state.ball.x = state.opponent.x; state.ball.y = state.opponent.y;
    } else {
      state.player.x = 50; state.player.y = PH - 20; state.player.hasBall = true;
      state.opponent.x = 50; state.opponent.y = 30; state.opponent.hasBall = false;
      state.ball.x = state.player.x; state.ball.y = state.player.y;
    }
    state.ball.vx = state.ball.vy = 0;
    state.running = true;
  }, 900);
 }

 // Timer & UI
 setInterval(()=>{
  if(state.running && state.timeLeft > 0){
    state.timeLeft--;
    if(state.timeLeft <= 0){
      state.running = false;
      setTimeout(()=>{ alert(`Match ended: You ${state.scoreYou} — Opp ${state.scoreOpp}`); },
150);
    }
    updateUI();
  }
 }, 1000);

 function updateUI(){
  timeEl.textContent = state.timeLeft;
  scoreYouEl.textContent = state.scoreYou;
  scoreOppEl.textContent = state.scoreOpp;
  powerBar.style.width = Math.max(6, Math.min(100, state.power)) + '%';
  modeEl.textContent = state.mode.charAt(0).toUpperCase() + state.mode.slice(1);
 }

 // Respawn & full reset
 function fullReset(){
  state.scoreYou = state.scoreOpp = 0; state.timeLeft = 60; state.running = true; updateUI();
resetRound();
 }

 // Main loop
 let last = performance.now();
 function step(now){
  const dt = Math.min(40, now - last) / 16.666;
```

```
    last = now;
    if(state.running){
      // keyboard movement
      let mx = 0, my = 0;
      if(keys['w'] || keys['arrowup']) my -= 1;
      if(keys['s'] || keys['arrowdown']) my += 1;
      if(keys['a'] || keys['arrowleft']) mx -= 1;
      if(keys['d'] || keys['arrowright']) mx += 1;
      if(mx !== 0 || my !== 0){
        state.player.vx += mx * 0.12 * state.player.speed;
        state.player.vy += my * 0.12 * state.player.speed;
      }
      // pointer drag movement (mobile)
      if(pointer.down && pointer.dragging && state.player.hasBall){
        const rect = canvas.getBoundingClientRect();
        const world = screenToWorldXY(pointer.x, pointer.y);
        const dx = world.x - state.player.x, dy = world.y - state.player.y;
        state.player.vx += dx * 0.04;
        state.player.vy += dy * 0.04;
      }

      // integrate players
      integrate(state.player, dt, state.player.speed);
      integrate(state.opponent, dt, state.opponent.speed);

      // update ball
      state.ball.x += state.ball.vx * dt;
      state.ball.y += state.ball.vy * dt;
      state.ball.vx *= 0.985; state.ball.vy *= 0.985;

      // possession pickups
      if(!state.player.hasBall && dist(state.ball, state.player) < 4){
        state.player.hasBall = true; state.opponent.hasBall = false;
        state.ball.vx = state.ball.vy = 0; state.ball.x = state.player.x; state.ball.y = state.player.y;
      }
      if(!state.opponent.hasBall && dist(state.ball, state.opponent) < 4){
        state.opponent.hasBall = true; state.player.hasBall = false;
        state.ball.vx = state.ball.vy = 0; state.ball.x = state.opponent.x; state.ball.y =
state.opponent.y;
      }

      clampEntity(state.player);
      clampEntity(state.opponent);
      clampBall();
```

```
    // AI
    updateAI(dt);

    // check goal lines (top is opponent goal; bottom is your goal)
    if(state.ball.y < 2){
      // opponent scored
      state.scoreOpp++;
      updateUI();
      respawnAfterGoal(false);
    } else if(state.ball.y > PH - 2){
      // player scored
      state.scoreYou++;
      updateUI();
      respawnAfterGoal(true);
    }
  }

  render();
  requestAnimationFrame(step);
}

// Render function
function render(){
  ctx.clearRect(0,0,W,H);
  // pitch gradient
  const g = ctx.createLinearGradient(0,0,W,H);
  g.addColorStop(0,'#0b6b3e'); g.addColorStop(1,'#0b5a36');
  ctx.fillStyle = g;
  ctx.fillRect(0,0,W,H);

  // boundary & lines
  ctx.strokeStyle = 'rgba(255,255,255,0.07)';
  ctx.lineWidth = 2;
  ctx.strokeRect(18,18,W-36,H-36);
  ctx.beginPath(); ctx.moveTo(W/2,18); ctx.lineTo(W/2,H-18); ctx.stroke();
  ctx.beginPath(); ctx.arc(W/2, H/2, 46, 0, Math.PI*2); ctx.stroke();

  // goals
  ctx.fillStyle = 'rgba(255,255,255,0.06)';
  ctx.fillRect(W*0.38,18-6, W*0.24, 6); ctx.fillRect(W*0.38, H-18, W*0.24, 6);

  // players
  drawPlayer(state.opponent, '#ff6b6b', 'AI');
```

```
    drawPlayer(state.player, '#00d1ff', 'YOU');

    // ball
    drawBall();

    // bottom overlay
    ctx.fillStyle = 'rgba(0,0,0,0.06)'; ctx.fillRect(0,H-62,W,62);
  }

  function drawPlayer(p, color, label){
    const x = vtx(p.x), y = vty(p.y);
    // shadow
    ctx.fillStyle = 'rgba(0,0,0,0.3)';
    ctx.beginPath(); ctx.ellipse(x, y+12, 18, 6, 0,0,Math.PI*2); ctx.fill();
    // body
    ctx.fillStyle = color;
    ctx.beginPath(); ctx.arc(x, y, Math.round(6 + p.radius/1.2), 0, Math.PI*2); ctx.fill();
    ctx.fillStyle = '#001'; ctx.font = '12px sans-serif';
    ctx.fillText(label === 'YOU' ? 'U' : 'A', x-5, y+4);
    if(p.hasBall){
      ctx.strokeStyle = 'rgba(255,255,255,0.12)'; ctx.lineWidth = 2;
      ctx.beginPath(); ctx.arc(x, y, 12, 0, Math.PI*2); ctx.stroke();
      // ball follows
      state.ball.x = p.x; state.ball.y = p.y;
      state.ball.vx = state.ball.vy = 0;
    }
  }

  function drawBall(){
    const bx = vtx(state.ball.x), by = vty(state.ball.y);
    ctx.fillStyle = '#fff'; ctx.beginPath(); ctx.arc(bx, by, 7, 0, Math.PI*2); ctx.fill();
    ctx.fillStyle = 'rgba(0,0,0,0.1)'; ctx.beginPath(); ctx.arc(bx-3, by-2, 2.5, 0, Math.PI*2); ctx.fill();
  }

  // quick power adjust: long-press on power bar area to charge
  let pTimer = null;
  powerBar.parentElement.parentElement.addEventListener('pointerdown', e=>{
    pTimer = setInterval(()=>{ state.power = clamp(state.power + 2, 6, 100); updateUI(); }, 90);
  });
  window.addEventListener('pointerup', e=>{ clearInterval(pTimer); pTimer = null; });

  // prevent touch scroll
  document.addEventListener('touchmove', e=>{ if(e.target === canvas ||
e.target.closest('#field')) e.preventDefault(); }, {passive:false});
```

```
  updateUI();
  requestAnimationFrame(step);
})();
</script>
</body>
</html>
```