<!--
https://mcorpai.org/

=== AI game Start ===

The PDF includes the full cognitive core of an 18 KB self-learning AI.

It uses reward-based learning and policy updates built directly in code.

The `AI_ARMS` array defines possible actions like jump or missile fire.

Success and failure are evaluated by the formula *s / (s + f)* — a classic bandit rule.

All learning and updates happen locally, with no network access.

Ethical control stops autonomy whenever human input is detected.

The AI instantly gives control back to the user — a coded moral layer.

Memory is stored in the browser so it recalls past survival performance.

It adapts from both short-term (success/failure) and long-term experience.

In essence, this 18 KB code is a functioning micro-brain that perceives, decides, and self-corrects offline.

-->

```
<style>
  :root {
    --rsr-track-h: 120px;
    --rsr-radius: 14px;
    --rsr-ground: 10px;
  }
  #rsr-root { font-family: system-ui, -apple-system, Segoe UI, Roboto, "Apple SD Gothic Neo", "Noto Sans", sans-serif; color:#111; display:flex; flex-direction:column; align-items:center; gap:8px; padding:16px 10px; }
  .rsr-title { font-weight:800; font-size:18px; letter-spacing:.2px; margin:0; }
```

```
  .rsr-hud { display:flex; gap:10px; flex-wrap:wrap; align-items:center;
font-size:13px; color:#1a1a1a; }
  .rsr-badge { padding:4px 8px; border-radius:999px; border:1px solid
#e5e7eb; background:#fafafa; }
  .rsr-temp strong{ font-weight:800; }
  .rsr-wrap { width:min(1100px, 100%); position:relative; }
  .rsr-runner { position:relative; width:100%; height:var(--rsr-track-h);
border:1px solid #e5e7eb; border-radius:var(--rsr-radius); overflow:hidden;
background:linear-gradient(180deg,#ffffff 0%, #f8fbff 100%); box-shadow:0
1px 0 rgba(0,0,0,.04), inset 0 -20px 40px rgba(0,0,0,.02); }
  #rsr-canvas { width:100%; height:100%; display:block; }
  .rsr-overlay { position:absolute; inset:0; background:rgba(255,255,255,.96);
display:none; align-items:center; justify-content:center; text-align:center;
padding:18px; }
  .rsr-overlay-inner{ max-width:560px; }
  .rsr-overlay h2{ margin:0 0 6px 0; font-size:18px; font-weight:800; }
  .rsr-overlay p{ margin:0 0 12px 0; font-size:14px; color:#333; }
  .rsr-btn { margin-top:8px; padding:9px 14px; border-radius:10px; border:1px
solid #e5e7eb; background:#111827; color:#fff; font-weight:700;
cursor:pointer; }
  .rsr-ground { position:absolute; left:0; right:0; bottom:0;
height:var(--rsr-ground); background:repeating-linear-gradient(90deg,#b6c1cd
0 22px,#c5cfda 22px 44px); pointer-events:none; }
  @media (max-width:600px){ :root{ --rsr-track-h: 100px; } .rsr-hud{
font-size:12px; } }
</style>

<script>
(function(){
  'use strict';

  function onReady(fn){ if(document.readyState==='loading')
document.addEventListener('DOMContentLoaded', fn, {once:true}); else fn(); }

  onReady(()=>{
    document.title = 'Refugee Self-Reliance';

    // --- DOM scaffold ---
    const root = document.createElement('div');
    root.id = 'rsr-root';
```

```
  root.innerHTML = `
<div class="rsr-title">
 Refugee Self-Reliance Sovereign AI 🌐
 <span style="color:#1e90ff;">
  This is an emoji-sized game (18 Kb) that contains a real-time micro offline AI.
 </span>
</div>


    <div class="rsr-hud">
      <span class="rsr-badge rsr-temp">🌡️ Climate Temperature <strong id="rsr-temp">1.5°C</strong></span>
      <span class="rsr-badge">Score <strong id="rsr-score">0</strong></span>
      <span class="rsr-badge">Hunger <strong id=\"rsr-hunger\">25.0s<\/strong></span>
      <span class="rsr-badge">Press Space to jump</span>
      <span class="rsr-badge" id="rsr-ai-msg" style="display:none"></span>
    </div>
    <div class="rsr-wrap">
      <div class="rsr-runner">
        <canvas id="rsr-canvas"></canvas>
        <div class="rsr-ground"></div>
        <div class="rsr-overlay" id="rsr-overlay" role="dialog" aria-modal="true">
          <div class="rsr-overlay-inner">
            <h2>Refugee Self-Reliance at mcorpai.org</h2>
            <p>Free public-interest partnerships available.</p>
            <p class="rsr-stats" style="margin-top:6px"></p>
            <button class="rsr-btn" id="rsr-retry" type="button">Retry</button>
          </div>
        </div>
      </div>

      <!-- PDF link (centered, small) placed directly under the game -->
      <div style="width:100%; display:flex; justify-content:center; margin-top:8px;">
        <a href="The_18KB_AI_Redefining_Intelligence_through_Ethics_and_Autonomy.
```

```
pdf" target="_blank" rel="noopener" style="font-size:12px; color:#1f2937;
text-decoration:underline;">
          The 18KB AI — Redefining Intelligence through Ethics and Autonomy
(PDF) 2025-10-31
        </a>
       </div>
     </div>`;
   if(document.body?.prepend) document.body.prepend(root); else
(document.body? document.body.insertBefore(root,
document.body.firstChild||null) :
document.documentElement.appendChild(root));

   // --- Refs ---
   const tempEl = document.getElementById('rsr-temp');
   const scoreEl = document.getElementById('rsr-score');
   const hungerEl = document.getElementById('rsr-hunger');
   const overlay = document.getElementById('rsr-overlay');
   const statsEl = overlay.querySelector('.rsr-stats');
   const retryBtn = document.getElementById('rsr-retry');
   const aiMsg = document.getElementById('rsr-ai-msg');

   // --- Canvas ---
   const canvas = document.getElementById('rsr-canvas');
   const ctx = canvas.getContext('2d');
   let W=0, H=0, DPR = Math.max(1, window.devicePixelRatio||1);

   // Tunables (~3 min target)
   const PLAYER_EMOJI = '🐱'; // change to '🐱' if preferred
   let speedBase = 300; const gravity = 2200; const jumpV = -800;

   // State
   let running = true, speedMul = 1, groundY = 0; let player = null;
   const obstacles=[], foods=[], peaceBursts=[], missiles=[], explosions=[];
   let score = 0, temp = 1.5, hunger = 25.0;
   let lastSpawnObs=0, spawnObsEvery=1.4, lastSpawnFood=0,
spawnFoodEvery=5.0;
   let lastJumpAt=-999, streak=0, foodsEaten=0; let startTime =
performance.now();

   // Shelter
```

```
  const shelter = { active:false, t:0, dur:3.2 }; let nextShelterIn = rand(10,14);
function scheduleShelter(){ nextShelterIn = rand(14,22);}

  // --- AI persistence ---
  const AI_KEY='rsr_ai_policy_v2';
  // Arms: both Jump(J) and Missile(M) with decision thresholds (seconds to
collision)
  const AI_ARMS=[

{type:'J',t:0.34},{type:'J',t:0.30},{type:'J',t:0.26},{type:'J',t:0.22},{type:'J',t:0.18},

{type:'M',t:0.34},{type:'M',t:0.28},{type:'M',t:0.24},{type:'M',t:0.20},{type:'M',t:0.1
6}
  ];
  let ai = loadPolicy();
  let aiActsLeft = chooseBudget(ai.prevSurvival||0); // 10..30
  let lastUserInput=-999; // sec
  let aiFireCooldown=0; // seconds between missiles
  let pendingEval = null; // {armIdx, t}

  function loadPolicy(){
    try{ const raw=localStorage.getItem(AI_KEY); if(raw){ const
o=JSON.parse(raw); if(o && Array.isArray(o.arms) &&
o.arms.length===AI_ARMS.length) return o; } }catch(e){}
    return { arms: AI_ARMS.map((a,i)=>({ key:a.type+'-'+a.t, type:a.type, t:a.t,
s:1, f:1 })), total:0, prevSurvival:0 };
  }
  function savePolicy(){ try{ localStorage.setItem(AI_KEY, JSON.stringify(ai));
}catch(e){} }

  function chooseBudget(prevSec){
    // Map last survival to assistance budget (10..30)
    if(prevSec<=30) return 30;
    if(prevSec<=60) return 26;
    if(prevSec<=90) return 22;
    if(prevSec<=120) return 18;
    if(prevSec<=180) return 14;
    return 10;
  }
```

```
    function pickArm(){
      const eps = ai.total < 60 ? 0.20 : 0.10;
      if(Math.random()<eps) return (Math.random()*AI_ARMS.length)|0;
      let best=-1, bestIdx=0;
      for(let i=0;i<ai.arms.length;i++){
        const a=ai.arms[i]; const mean=a.s/(a.s+a.f); if(mean>best){ best=mean;
bestIdx=i; }
      }
      return bestIdx;
    }
    function updateArm(idx, success){ const a=ai.arms[idx]; if(!a) return;
success? a.s++ : a.f++; ai.total++; savePolicy(); }

    function toastAI(){ aiMsg.textContent='A living example of ethical autonomy
— an AI that learns and decides for survival, entirely offline.';
aiMsg.style.display='inline-block'; clearTimeout(toastAI._t);
toastAI._t=setTimeout(()=>aiMsg.style.display='none',1600); }

    function resize(){
      const wrap=canvas.parentElement||root; W=wrap.clientWidth|0;
H=Math.max(90,Math.min(140,Math.floor(W*0.09)));
DPR=Math.max(1,window.devicePixelRatio||1);
      canvas.width=Math.floor(W*DPR); canvas.height=Math.floor(H*DPR);
canvas.style.width=W+'px'; canvas.style.height=H+'px';
ctx.setTransform(DPR,0,0,DPR,0,0);
      groundY=H-18; const size=Math.floor(H*0.36);
      if(!player){ player={ x:Math.floor(W*0.12), y:groundY-size, w:size*0.72,
h:size, vy:0, size, emoji:'🐱' }; }
      else { player.size=size; player.w=size*0.72; player.h=size;
player.y=groundY-player.h; }
    }
    window.addEventListener('resize',resize); resize();

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

    function spawnObstacle(){ const list=['🔫','💣','🧨','🪖','🧱','🛡️']; const
em=list[(Math.random()*list.length)|0]; const
hMul=Math.random()<0.08?1.1:1.0; const size=Math.floor(H*(0.24*hMul));
const w=Math.max(18,Math.floor(size*0.7)); const h=Math.max(22,size); const
```

```
x=W+30; const y=groundY-h+4;
obstacles.push({x,y,w,h,emoji:em,size:h,awarded:false}); }
  function spawnFood(){ const list=['🍞','🥫','🍚','🥖','🍎']; const
em=list[(Math.random()*list.length)|0]; const size=Math.floor(H*0.28); const
x=W+30; const lift=Math.min(40,H*0.3); const
y=groundY-size-Math.random()*lift-6;
foods.push({x,y,w:size*0.7,h:size,emoji:em,size}); }

  function rectsIntersect(a,b){ return a.x<b.x+b.w && a.x+a.w>b.x &&
a.y<b.y+b.h && a.y+a.h>b.y; }
  function clamp(v,lo,hi){ return Math.max(lo,Math.min(hi,v)); }

  function die(){
    running=false; streak=0; overlay.style.display='flex'; const
secs=((performance.now()-startTime)/1000).toFixed(1);
statsEl.textContent=`Final temp ${temp.toFixed(1)}°C · Survival ${secs}s ·
Food ${foodsEaten}`;
    if(pendingEval && (performance.now()*0.001 - pendingEval.t)<1.0){
updateArm(pendingEval.idx,false); pendingEval=null; }
    ai.prevSurvival=parseFloat(secs); savePolicy();
  }

  function reset(){
    running=true; obstacles.length=0; foods.length=0; peaceBursts.length=0;
missiles.length=0; explosions.length=0;
    score=0; temp=1.5; hunger=20.0; speedMul=1; lastSpawnObs=0;
spawnObsEvery=1.0; lastSpawnFood=0; spawnFoodEvery=6.0;
    player.vy=0; player.y=groundY-player.h; streak=0; foodsEaten=0;
startTime=performance.now();
    shelter.active=false; shelter.t=0; scheduleShelter();
aiActsLeft=chooseBudget(ai.prevSurvival||0); aiFireCooldown=0;
pendingEval=null; aiMsg.style.display='none';
    overlay.style.display='none'; lastTime=performance.now();
requestAnimationFrame(loop);
  }

  // Input
  window.addEventListener('keydown',(e)=>{ if(e.code==='Space'){
e.preventDefault(); if(!running) return; const
onGround=player.y>=groundY-player.h-0.5; if(onGround){ player.vy=jumpV;
```

```
lastJumpAt=performance.now()*0.001; lastUserInput=lastJumpAt; } }
if(e.code==='KeyR'){ e.preventDefault(); reset(); } },{passive:false});
    canvas.addEventListener('pointerdown',()=>{ const
onGround=player.y>=groundY-player.h-0.5; if(running && onGround){
player.vy=jumpV; lastJumpAt=performance.now()*0.001;
lastUserInput=lastJumpAt; } });
    retryBtn.addEventListener('click',reset);

    // Peace bubble
    function burstPeace(){
peaceBursts.push({t:0,life:0.6,x:player.x+player.w*0.6,y:player.y-10});
if(peaceBursts.length>6) peaceBursts.shift(); }

    // Missile helper
    function fireMissile(){
      const size=Math.floor(H*0.22); const y=player.y+player.h - size - 4;
missiles.push({ x:player.x+player.w*0.6, y, w:size*0.8, h:size, size, emoji:'🚀',
vx: 1050, born: performance.now()*0.001 }); aiFireCooldown=0.35; }

    // AI controller
    function aiController(dt, speed){
      if(aiActsLeft<=0 || !running) return; const now=performance.now()*0.001;
if(now-lastUserInput<0.12) return; if(aiFireCooldown>0){ aiFireCooldown-=dt; }
      // nearest obstacle
      let nearest=null, dist=Infinity; for(const o of obstacles){ const
d=(o.x-(player.x+player.w)); if(d>0 && d<dist){ dist=d; nearest=o; } }
      if(!nearest) return; const ttc=dist/speed; const idx=pickArm(); const
arm=ai.arms[idx]; const onGround=player.y>=groundY-player.h-0.5;
      if(arm.type==='J' && onGround && ttc<arm.t){ player.vy=jumpV;
lastJumpAt=now; lastUserInput=now; aiActsLeft--; toastAI(); pendingEval={
idx, t: now }; }
      else if(arm.type==='M' && aiFireCooldown<=0 && ttc<arm.t){ fireMissile();
aiActsLeft--; toastAI(); pendingEval={ idx, t: now, missile:true }; }
      // mark success if survived ~0.9s after AI act and no failure was recorded
      if(pendingEval && (now-pendingEval.t)>0.9 && !pendingEval.waitHit){
updateArm(pendingEval.idx,true); pendingEval=null; }
    }

    // Loop
    let lastTime=performance.now();
```

```
  function loop(nowMs){
    const now=nowMs*0.001; let dt=(nowMs-lastTime)/1000; if(dt>0.05)
dt=0.05; lastTime=nowMs; if(!running) return;

    // Shelter
    if(!shelter.active){ nextShelterIn-=dt; if(nextShelterIn<=0){
shelter.active=true; shelter.t=0; temp=clamp(temp-0.10,1.5,3.0);} } else {
shelter.t+=dt; if(shelter.t>=shelter.dur){ shelter.active=false; scheduleShelter();
} }

    // Climate
    const baseRise=0.0032;
temp=clamp(temp+(shelter.active?-0.020:baseRise)*dt*60,1.5,3.0); const
diff=(temp-1.5)/1.5;

    // Speed
    const elapsed=(performance.now()-startTime)/1000; const
ramp=Math.min(1,elapsed/12); speedMul=1+diff*0.4; const
speed=(speedBase*(0.7+0.3*ramp))*speedMul;

    // Spawns
    spawnObsEvery=clamp(1.60 - diff*0.25, 0.90, 1.60);
spawnFoodEvery=clamp(4.5 - diff*1.5, 2.0, 4.5);
    hunger-=dt; if(hunger<0) hunger=0; if(hunger===0){ die(); }

    player.vy+=gravity*dt; player.y+=player.vy*dt;
if(player.y>groundY-player.h){ player.y=groundY-player.h; player.vy=0; }

    lastSpawnObs+=dt; if(!shelter.active &&
lastSpawnObs>=spawnObsEvery){ lastSpawnObs=0; spawnObstacle(); const
allowDouble=elapsed>8; if(allowDouble && Math.random()<(0.02+diff*0.05))
setTimeout(spawnObstacle,Math.floor(rand(150,300))); }
    lastSpawnFood+=dt; if(lastSpawnFood>=spawnFoodEvery){
lastSpawnFood=0; spawnFood(); } if(hunger<12 && foods.length===0){
spawnFood(); }

    // AI decide before movement integration of entities
    aiController(dt, speed);

    // Move obstacles & collisions
```

```
    for(let i=obstacles.length-1;i>=0;i--){ const o=obstacles[i]; o.x-=speed*dt;
if(rectsIntersect({x:player.x,y:player.y,w:player.w,h:player.h},{x:o.x,y:o.y,w:o.w,h
:o.h})){ die(); }
      if(!o.awarded && o.x+o.w<player.x-player.w*0.1){ o.awarded=true;
if(now-lastJumpAt<0.9){ streak=Math.min(streak+1,10); burstPeace(); } }
      if(o.x+o.w<-20) obstacles.splice(i,1); }

    for(let i=foods.length-1;i>=0;i--){ const f=foods[i]; f.x-=speed*dt*0.98;
if(rectsIntersect({x:player.x,y:player.y,w:player.w,h:player.h},{x:f.x,y:f.y,w:f.w,h:f.
h})){ foods.splice(i,1); score+=50; hunger=20.0; foodsEaten++; } else
if(f.x+f.w<-20){ foods.splice(i,1);} }

    // Missiles
    for(let i=missiles.length-1;i>=0;i--){ const m=missiles[i]; m.x+= (m.vx +
speed*0.2) * dt; // outrun obstacles
      // hit check
      let hit=false; for(let j=obstacles.length-1;j>=0;j--){ const o=obstacles[j];
if(rectsIntersect({x:m.x,y:m.y,w:m.w,h:m.h},{x:o.x,y:o.y,w:o.w,h:o.h})){
obstacles.splice(j,1); hit=true; break; } }
      if(hit){ explosions.push({x:m.x+ m.w*0.5, y:m.y+ m.h*0.4, t:0, life:0.35});
missiles.splice(i,1); if(pendingEval && pendingEval.missile){
updateArm(pendingEval.idx,true); pendingEval=null; } continue; }
      if(m.x > W+60){ missiles.splice(i,1); }
    }

    // Explosions fade
    for(let i=explosions.length-1;i>=0;i--){ const e=explosions[i]; e.t+=dt;
if(e.t>e.life) explosions.splice(i,1); }

    // Score
    score += dt * (10 + 6 * diff) * speedMul + streak * dt;

    // Draw
    ctx.clearRect(0,0,W,H);
    if(shelter.active){ ctx.globalAlpha=0.12; ctx.fillStyle='#22c55e';
ctx.fillRect(0,0,W,H); ctx.globalAlpha=1; ctx.font="12px system-ui,
-apple-system, 'Noto Sans'"; ctx.fillStyle='#14532d'; ctx.fillText('Shelter',10,16);
}
    ctx.strokeStyle='#cdd6df'; ctx.lineWidth=2; ctx.beginPath();
ctx.moveTo(0,groundY+6); ctx.lineTo(W,groundY+6); ctx.stroke();
```

```
    for(const f of foods){ ctx.font=Math.floor(f.size)+"px 'Apple Color Emoji','Segoe UI Emoji','Noto Color Emoji',sans-serif"; ctx.textBaseline='bottom'; ctx.fillText(f.emoji,f.x,f.y+f.h); }
    for(const o of obstacles){ ctx.font=Math.floor(o.size)+"px 'Apple Color Emoji','Segoe UI Emoji','Noto Color Emoji',sans-serif"; ctx.textBaseline='bottom'; ctx.fillText(o.emoji,o.x,o.y+o.h); }
    // missiles
    for(const m of missiles){ ctx.font=Math.floor(m.size)+"px 'Apple Color Emoji','Segoe UI Emoji','Noto Color Emoji',sans-serif"; ctx.textBaseline='bottom'; ctx.fillText('🚀', m.x, m.y + m.h); }
    for(const e of explosions){ const a=1-(e.t/e.life); ctx.globalAlpha=Math.max(0,a); ctx.font=Math.floor(H*0.30)+"px 'Apple Color Emoji','Segoe UI Emoji','Noto Color Emoji',sans-serif"; ctx.fillText('💥', e.x, e.y); ctx.globalAlpha=1; }
    // player
    ctx.font=Math.floor(player.size)+"px 'Apple Color Emoji','Segoe UI Emoji','Noto Color Emoji',sans-serif"; ctx.textBaseline='bottom'; ctx.fillText(player.emoji, player.x, player.y+player.h);
    // peace bubbles
    for(let i=peaceBursts.length-1;i>=0;i--){ const b=peaceBursts[i]; b.t+=dt; const a=1-(b.t/b.life); if(a<=0){ peaceBursts.splice(i,1); continue;} const y=b.y-(b.t*28); const text='Peace'; ctx.globalAlpha=Math.max(0,a*0.95); const sizeBoost=Math.min(4,streak*0.2); ctx.font=(11+sizeBoost)+"px system-ui, -apple-system, 'Noto Sans', sans-serif"; const tw=ctx.measureText(text).width+10; const tx=b.x, ty=y; ctx.fillStyle='#ffffff'; ctx.strokeStyle='#d9e0e7'; ctx.lineWidth=1; roundRect(ctx, tx, ty, tw, 18+sizeBoost*0.6, 9, true, true); ctx.fillStyle='#222'; ctx.fillText(text, tx+5, ty+13); ctx.globalAlpha=1; }

    // HUD
    tempEl.textContent= temp.toFixed(1)+'°C'; hungerEl.textContent=hunger.toFixed(1)+'s'; scoreEl.textContent=Math.floor(score).toLocaleString();

    requestAnimationFrame(loop);
  }

  function roundRect(ctx,x,y,w,h,r,fill,stroke){ if(w<2*r) r=w/2; if(h<2*r) r=h/2; ctx.beginPath(); ctx.moveTo(x+r,y); ctx.arcTo(x+w,y,x+w,y+h,r);
```

```
ctx.arcTo(x+w,y+h,x,y+h,r); ctx.arcTo(x,y+h,x,y,r); ctx.arcTo(x,y,x+w,y,r);
ctx.closePath(); if(fill) ctx.fill(); if(stroke) ctx.stroke(); }

    // Self-tests
    console.groupCollapsed('RSR self-tests');
    console.assert(document.getElementById('rsr-root'),'root exists');
    console.assert(document.getElementById('rsr-canvas'),'canvas exists');
    console.assert(!!ctx,'2D context');
    console.assert(document.getElementById('rsr-overlay'),'overlay exists');
    console.assert(document.getElementById('rsr-retry'),'retry exists');
    console.assert(Array.isArray(AI_ARMS) && AI_ARMS.length>=6,'AI arms
ready');
    console.assert(typeof aiActsLeft==='number' && aiActsLeft>=10 &&
aiActsLeft<=30,'AI budget 10..30');
    console.groupEnd();

    requestAnimationFrame(loop);

  });
})();

</script>

<!-- === AI game End === -->
```