```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Bee Swarm Clicker - Fixes</title>
<style>
 body {
   margin: 0;
   height: 100vh;
   background: linear-gradient(to bottom, #87CEEB, #E0F7FA);
   font-family: 'Fredoka One', 'Comic Sans MS', cursive;
   overflow: hidden;
   user-select: none;
 }

 #game {
   width: 100vw;
   height: 100vh;
   display: flex;
   align-items: center;
   position: relative;
 }

 #field {
   width: 560px;
   height: 560px;
   background: linear-gradient(to bottom, #98FB98, #90EE90);
   border-radius: 30px;
   border: 18px solid #228B22;
   box-shadow: inset 0 0 90px rgba(0,0,0,0.25), 0 30px 60px rgba(0,0,0,0.4);
   margin-left: 40px;
   position: relative;
   overflow: hidden;
 }

 /* Flower Styling */
 .flower-square {
   width: 70px;
   height: 70px;
   position: absolute;
   border-radius: 12px;
   display: flex;
   flex-wrap: wrap;
   align-items: center;
```

```
  justify-content: center;
  gap: 3px;
  cursor: pointer;
  transition: transform 0.05s ease;
  box-shadow: 0 4px 8px rgba(0,0,0,0.2);
  background: rgba(255,255,255,0.5);
}

.flower-square:active { transform: scale(0.95); }

/* Flower Colors */
.flower-square.white  { background: linear-gradient(135deg, #ffffff, #e0e0e0) !important; }
.flower-square.blue   { background: linear-gradient(135deg, #64b5f6, #1976d2) !important; }
.flower-square.red    { background: linear-gradient(135deg, #ff8a80, #d32f2f) !important; }
.flower-square.orange { background: linear-gradient(135deg, #FFB74D, #F57C00) !important; }

.flower { font-size: 26px; pointer-events: none; }

.top-bar {
  position: absolute;
  top: 16px;
  left: 50%;
  transform: translateX(-50%);
  display: flex;
  gap: 20px;
  z-index: 100;
}

.top-indicator {
  background: rgba(255,255,255,0.92);
  padding: 8px 20px;
  border-radius: 30px;
  font-size: 22px;
  font-weight: bold;
  box-shadow: 0 6px 16px rgba(0,0,0,0.2);
  backdrop-filter: blur(6px);
  border: 3px solid #FFD700;
  color: #333;
}

#honey-display { background: linear-gradient(135deg, #FFD700, #FFB600); color: #8B4513; }

#bag-display {
  position: absolute;
```

```
  bottom: 30px;
  left: 50%;
  transform: translateX(-50%);
  padding: 12px 32px;
  border-radius: 30px;
  font-size: 28px;
  font-weight: bold;
  box-shadow: 0 8px 20px rgba(0,0,0,0.3);
  border: 4px solid #FFA500;
  color: black;
  transition: background 0.3s ease;
}

#convert-btn {
  position: absolute;
  bottom: 40px;
  left: 40px;
  padding: 16px 38px;
  font-size: 26px;
  background: linear-gradient(to bottom, #FFD700, #FFB600);
  color: #8B4513;
  border: none;
  border-radius: 50px;
  cursor: pointer;
  box-shadow: 0 10px #B8860B;
  font-weight: bold;
}

#convert-btn:hover { background: linear-gradient(to bottom,#FFEA80,#FFDB4D); }
#convert-btn:active { transform: translateY(6px); box-shadow: 0 4px #B8860B; }

.pollen-popup {
  position: absolute;
  font-weight: bold;
  font-size: 38px;
  color: #FFD700;
  text-shadow: 3px 3px 8px #000;
  pointer-events: none;
  animation: float 1.3s ease-out forwards;
  z-index: 999;
}

@keyframes float {
  0% { opacity:1; transform:translateY(0) scale(1); }
```

```
  100% { opacity:0; transform:translateY(-130px) scale(0.7); }
}

#field-dropdown {
  position: absolute;
  top: 60px;
  left: 50%;
  transform: translateX(-50%);
  background: rgba(255,255,255,0.95);
  border-radius: 20px;
  box-shadow: 0 8px 24px rgba(0,0,0,0.25);
  padding: 12px 0;
  z-index: 200;
  display: none;
  min-width: 240px;
  max-height: 80vh;
  overflow-y: auto;
  border: 4px solid #FFD700;
}

.dropdown-item {
  padding: 10px 24px;
  font-size: 18px;
  cursor: pointer;
  transition: background 0.2s;
  text-align: center;
  border-bottom: 1px solid #eee;
}

.dropdown-item:hover { background: #FFD700; color: #8B4513; }

#shop-toggle {
  position: absolute;
  right: 20px;
  top: 38%;
  padding: 15px 25px;
  background: linear-gradient(to bottom, #FFD700, #FFB600);
  color: #8B4513;
  border: none;
  border-radius: 30px;
  cursor: pointer;
  font-size: 24px;
  font-weight: bold;
  box-shadow: 0 8px 20px rgba(0,0,0,0.3);
```

```
  z-index: 150;
}

#hive-btn {
  position: absolute;
  left: calc(50% - 420px);
  top: 18px;
  transform: translateX(-50%);
  padding: 10px 16px;
  background: linear-gradient(to bottom, #FFDE7D, #FFD14A);
  border-radius: 12px;
  border: 3px solid #B8860B;
  cursor: pointer;
  z-index: 160;
  font-weight: bold;
}

#shop-panel {
  position: absolute;
  right: -380px;
  top: 0;
  width: 380px;
  height: 100vh;
  background: rgba(255,255,255,0.98);
  border-left: 4px solid #FFD700;
  box-shadow: -10px 0 30px rgba(0,0,0,0.2);
  transition: right 0.3s ease;
  padding: 20px;
  overflow-y: auto;
  z-index: 100;
}

#shop-panel.open { right: 0; }

.shop-section { margin-bottom: 30px; padding-bottom: 20px; border-bottom: 2px solid #ddd; }
.shop-section h3 { color: #8B4513; text-align: center; margin-bottom: 15px; }

.item {
  background: #f9f9f9;
  border: 2px solid #ddd;
  border-radius: 15px;
  padding: 15px;
  margin-bottom: 15px;
  cursor: pointer;
```

```
    transition: all 0.2s;
  }
  .item:hover { border-color: #FFD700; box-shadow: 0 4px 12px rgba(255,215,0,0.3); }
  .item.bought { background: #e0e0e0; cursor: default; }

  .item-name { font-size: 18px; font-weight: bold; margin-bottom: 5px; }
  .item-stats { font-size: 13px; color: #666; margin-bottom: 5px; }
  .item-cost { font-size: 16px; color: #FFD700; font-weight: bold; }

  .buy-btn { width: 100%; padding: 8px; background: #FFD700; color: #8B4513; border: none;
border-radius: 10px; font-weight: bold; cursor: pointer; }
  .buy-btn:disabled { background: #ccc; cursor: not-allowed; }

  /* Inventory & Hive */
  #inventory-toggle {
    position: absolute;
    right: 20px;
    top: 55%;
    padding: 12px 20px;
    background: linear-gradient(to bottom, #FFF1A8, #FFE08A);
    color: #8B4513;
    border: none;
    border-radius: 30px;
    cursor: pointer;
    font-size: 18px;
    font-weight: bold;
    box-shadow: 0 8px 20px rgba(0,0,0,0.25);
    z-index: 150;
  }

  #inventory-panel {
    position: absolute;
    right: -350px;
    top: 0;
    width: 350px;
    height: 100vh;
    background: rgba(255,255,255,0.96);
    border-left: 4px solid #FFD700;
    box-shadow: -10px 0 30px rgba(0,0,0,0.2);
    transition: right 0.3s ease;
    padding: 12px;
    overflow-y: auto;
    z-index: 101;
  }
```

```
  #inventory-panel.open { right: 0; }

  .inv-tabs { display:flex; gap:6px; margin-bottom:12px; }
  .inv-tab { flex:1; padding:8px 6px; text-align:center; cursor:pointer; border-radius:8px;
border:2px solid #FFD700; background:#fff; font-weight:bold; }
  .inv-tab.active { background:#FFD700; color:#8B4513; }
  .inventory-grid { display:flex; gap:8px; flex-wrap:wrap; padding:8px; }
  .inv-item { width: 70px; height: 70px; border-radius:10px; background:#fff; border:2px solid
#ddd; display:flex; align-items:center; justify-content:center; font-weight:bold; cursor:grab; }

  #hive-view {
    position: absolute;
    left: calc(50% - 190px);
    top: 50%;
    transform: translateY(-50%);
    width: 380px;
    height: 760px;
    background: linear-gradient(180deg,#F2D9B3,#F8E7C4);
    border-radius: 18px;
    padding: 18px;
    box-shadow: 0 30px 60px rgba(0,0,0,0.4);
    z-index: 300;
    display: none;
  }
  #hive-header { display:flex; justify-content:space-between; align-items:center;
margin-bottom:12px; }
  #hive-grid { width: 100%; height: calc(100% - 48px); display:grid; grid-template-columns:
repeat(5,1fr); grid-auto-rows: 68px; gap:8px; overflow:auto; }
  .hive-slot { background: rgba(255,255,255,0.9); border-radius:8px; border:2px dashed
#c59b2a; display:flex; align-items:center; justify-content:center; font-weight:bold;
position:relative; }
  .hive-slot.empty { color:#8B4513; }
  .hive-slot .bee-icon { font-size:22px; pointer-events:none; }

  /* Quest Panel */
  #quest-toggle {
    position: absolute;
    right: 20px;
    top: 21%;
    padding: 15px 25px;
    background: linear-gradient(to bottom, #606060, #404040);
    color: #FFD700;
    border: none;
    border-radius: 30px;
```

```
    cursor: pointer;
    font-size: 24px;
    font-weight: bold;
    box-shadow: 0 8px 20px rgba(0,0,0,0.3);
    z-index: 150;
  }
  #quest-panel {
    position: absolute;
    right: -360px;
    top: 0;
    width: 360px;
    height: 100vh;
    background: rgba(40, 40, 40, 0.95);
    border-left: 4px solid #FFD700;
    box-shadow: -10px 0 30px rgba(0,0,0,0.4);
    transition: right 0.3s ease;
    padding: 20px;
    overflow-y: auto;
    z-index: 102;
    color: white;
  }
  #quest-panel.open { right: 0; }
  .quest-title { color: #FFD700; font-size: 22px; margin-bottom: 15px; text-align: center;
border-bottom: 2px solid #FFD700; padding-bottom: 5px; }
  .quest-item { margin-bottom: 20px; padding: 10px; border: 1px solid #444; border-radius: 8px;
background: rgba(255,255,255,0.1); }
  .quest-name { font-weight: bold; margin-bottom: 5px; font-size: 18px; color: #ffd700; }
  .quest-progress { font-size: 14px; margin-bottom: 5px; color: #eee; }
  .quest-subtask { font-size: 13px; margin-left: 10px; color: #ccc; }
  .quest-reward { font-size: 14px; color: #aaffaa; margin-top: 5px; font-weight: bold; }
  .quest-complete-btn { width: 100%; padding: 8px; margin-top: 10px; background: #28a745;
color: white; border: none; border-radius: 5px; cursor: pointer; font-weight: bold; }
  .quest-complete-btn:disabled { background: #555; cursor: not-allowed; }
  .bees-list { padding:8px; border-top:1px solid #ddd; margin-top:10px; }
  .bee-row { display:flex; justify-content:space-between; align-items:center; padding:6px 0;
border-bottom:1px dashed #eee; }
</style>
<link href="https://fonts.googleapis.com/css2?family=Fredoka+One&display=swap"
rel="stylesheet">
</head>
<body>

<div id="game">
  <div id="field"></div>
```

```
  <div class="top-bar">
    <div id="hive-btn" title="Open Hive">🏠 Hive</div>
    <div class="top-indicator" id="field-indicator">Dandy Field</div>
    <div class="top-indicator" id="honey-display">Honey: <span id="honey">0</span></div>
  </div>

  <div id="field-dropdown">
    <div class="dropdown-item" data-field="dandy">Dandy</div>
    <div class="dropdown-item" data-field="sunflower">Sunflower</div>
    <div class="dropdown-item" data-field="mushroom">Mushroom</div>
    <div class="dropdown-item" data-field="blueflower">Blue Flower</div>
    <div class="dropdown-item" data-field="clover">Clover</div>
    <div class="dropdown-item" data-field="bamboo">Bamboo (5 Bees)</div>
    <div class="dropdown-item" data-field="spider">Spider (5 Bees)</div>
    <div class="dropdown-item" data-field="strawberry">Strawberry (5 Bees)</div>
    <div class="dropdown-item" data-field="pineapple">Pineapple Patch (10 Bees)</div>
    <div class="dropdown-item" data-field="stump">Stump Field (10 Bees)</div>
    <div class="dropdown-item" data-field="cactus">Cactus Field (15 Bees)</div>
    <div class="dropdown-item" data-field="pumpkin">Pumpkin Patch (15 Bees)</div>
    <div class="dropdown-item" data-field="pine">Pine Tree Forest (15 Bees)</div>
    <div class="dropdown-item" data-field="rose">Rose Field (15 Bees)</div>
  </div>

  <div id="bag-display">Bag: <span id="current-bag">0</span> / <span
id="max-bag">200</span></div>
  <button id="convert-btn">Make Honey</button>

  <button id="quest-toggle">🐻 Quests</button>
  <button id="shop-toggle">🛒 Shop</button>
  <button id="inventory-toggle">🎒 Inventory</button>

  <div id="quest-panel">
    <div class="quest-title">Black Bear's Quests</div>
    <div id="quest-list"></div>
  </div>

  <div id="shop-panel">
    <h2 style="text-align: center; color: #8B4513;">Shop</h2>
    <div class="shop-section">
      <h3>Tools</h3>
      <div id="tools-list"></div>
    </div>
    <div class="shop-section">
```

```
      <h3>Backpacks</h3>
      <div id="backpacks-list"></div>
    </div>
    <div class="shop-section">
      <h3>Eggs</h3>
      <div id="eggs-list"></div>
    </div>
  </div>

  <div id="inventory-panel">
    <div style="text-align:center; font-weight:bold; color:#8B4513;
margin-bottom:8px;">Inventory</div>
    <div class="inv-tabs">
      <div class="inv-tab active" data-tab="items">Items</div>
      <div class="inv-tab" data-tab="bees">Bees</div>
    </div>
    <div id="items-tab" class="inv-content">
      <div style="font-size:14px; color:#666; padding:6px 2px;">Drag eggs from here into your
hive to hatch.</div>
      <div class="inventory-grid" id="inventory-grid"></div>
    </div>
    <div id="bees-tab" class="inv-content" style="display:none;">
      <div style="font-size:14px; color:#666; padding:6px 2px;">Your bees are collecting
pollen...</div>
      <div class="bees-list" id="bees-list"></div>
    </div>
  </div>

  <div id="hive-view">
    <div id="hive-header">
      <div style="font-weight:bold; font-size:20px; color:#8B4513;">Hive (5 x 10)</div>
      <button id="close-hive" style="padding:6px 10px; border-radius:8px; background:#ffd966;
border:2px solid #b8860b; cursor:pointer;">Close</button>
    </div>
    <div id="hive-grid"></div>
  </div>

</div>

<script>
// ========== Game state ==========
let pollen = 0;
let honey = 100000000;
let currentBag = 0;
```

```
let bagCapacity = 200;
let currentField = 'dandy';
let currentTool = 'hands';
let toolsBought = { hands: true };
let backpacksBought = { pouch: true };

// Inventory and Hive
let eggsInventory = { basic: 0 };
let eggsBoughtCount = { basic: 0 };
const BASIC_EGG_PRICES =
[100,250,625,1563,3906,9766,24414,61035,152588,381469,953674,2384186,5960464,149011
61,37252903,93132257,232830643,582076608,1455191520,3637978800,10000000000];

// Hive: 5 columns x 10 rows => 50 slots
const HIVE_COLS = 5;
const HIVE_ROWS = 10;
const hiveSlots = Array.from({length: HIVE_ROWS * HIVE_COLS}, () => null);
const BEES = [];

const BEE_STATS = {
  basic: { name: 'Basic Bee', pollenPerTick: 10, tickSeconds: 2, icon: '🐝' }
};

// UI elements
const fieldEl = document.getElementById('field');
const honeySpan = document.getElementById('honey');
const currentBagSpan = document.getElementById('current-bag');
const maxBagSpan = document.getElementById('max-bag');
const bagDisplay = document.getElementById('bag-display');
const fieldIndicator = document.getElementById('field-indicator');
const dropdown = document.getElementById('field-dropdown');
const convertBtn = document.getElementById('convert-btn');
const shopToggle = document.getElementById('shop-toggle');
const shopPanel = document.getElementById('shop-panel');
const toolsList = document.getElementById('tools-list');
const backpacksList = document.getElementById('backpacks-list');
const inventoryToggle = document.getElementById('inventory-toggle');
const inventoryPanel = document.getElementById('inventory-panel');
const inventoryGrid = document.getElementById('inventory-grid');
const eggsList = document.getElementById('eggs-list');
const invTabs = document.querySelectorAll('.inv-tab');
const beesListEl = document.getElementById('bees-list');
const hiveBtn = document.getElementById('hive-btn');
const hiveView = document.getElementById('hive-view');
```

```
const hiveGrid = document.getElementById('hive-grid');
const closeHive = document.getElementById('close-hive');

// Quest elements
const questToggle = document.getElementById('quest-toggle');
const questPanel = document.getElementById('quest-panel');
const questListEl = document.getElementById('quest-list');

const gridSize = 8;
const cellSize = 70;
const FLOWER_DURABILITY = 10;
const FLOWER_RESPAWN_TIME = 5000;
let flowerState = Array.from({length: gridSize}, () => Array(gridSize).fill(null));

// ========== Quests ==========
const BLACK_BEAR_QUESTS = [
  { id: 1, name: "Sunflower Start", description: "Collect 100 Pollen from the Sunflower Field.",
objective: { type: 'pollen', field: 'sunflower', amount: 100 }, reward: { honey: 200 }, rewardText:
"200 Honey" },
  { id: 2, name: "Dandelion Deed", description: "Collect 250 Pollen from the Dandy Field.",
objective: { type: 'pollen', field: 'dandy', amount: 250 }, reward: { honey: 400 }, rewardText: "400
Honey" },
  { id: 3, name: "Pollen Fetcher", description: "Collect 500 Pollen.", objective: { type:
'pollen_total', amount: 500 }, reward: { honey: 500 }, rewardText: "500 Honey" },
  { id: 4, name: "Red Request", description: "Collect 600 Red Pollen.", objective: { type:
'pollen_color', color: 'red', amount: 600 }, reward: { honey: 650 }, rewardText: "650 Honey" },
  { id: 5, name: "Into The Blue", description: "Collect 1000 Blue Pollen.", objective: { type:
'pollen_color', color: 'blue', amount: 1000 }, reward: { honey: 750 }, rewardText: "750 Honey" },
  { id: 6, name: "Variety Fetcher", description: "Collect pollen from various fields.", type: 'multi',
tasks: [{ text: "1,000 from Mushroom", type: 'pollen', field: 'mushroom', amount: 1000 }, { text:
"1,000 from Blue Flower", type: 'pollen', field: 'blueflower', amount: 1000 }, { text: "1,000 from
Clover", type: 'pollen', field: 'clover', amount: 1000 }], reward: { honey: 1000 }, rewardText:
"1,000 Honey, 1 Silver Egg" },
  { id: 7, name: "Bamboo Boogie", description: "Collect 4,000 Pollen from the Bamboo Field.",
objective: { type: 'pollen', field: 'bamboo', amount: 4000 }, reward: { honey: 2100 }, rewardText:
"2,100 Honey" },
  { id: 8, name: "Red Request 2", description: "Collect 5,000 Red Pollen.", objective: { type:
'pollen_color', color: 'red', amount: 5000 }, reward: { honey: 2500 }, rewardText: "2,500 Honey" },
  { id: 9, name: "Cobweb Sweeper", description: "Collect 6,000 Pollen from the Spider Field.",
objective: { type: 'pollen', field: 'spider', amount: 6000 }, reward: { honey: 3200 }, rewardText:
"3,200 Honey" },
  { id: 10, name: "Leisure Loot", description: "Collect 3,000 Pollen from multiple fields.", type:
'multi', tasks: [{ text: "3,000 from Dandy", type: 'pollen', field: 'dandy', amount: 3000 }, { text:
"3,000 from Sunflower", type: 'pollen', field: 'sunflower', amount: 3000 }, { text: "3,000 from
```

```
Mushroom", type: 'pollen', field: 'mushroom', amount: 3000 }, { text: "3,000 from Blue Flower", type: 'pollen', field: 'blueflower', amount: 3000 }], reward: { honey: 10000 }, rewardText: "10,000 Honey" },
  { id: 11, name: "White Pollen Wrangler", description: "Collect 10,000 White Pollen.", objective: { type: 'pollen_color', color: 'white', amount: 10000 }, reward: { honey: 7500 }, rewardText: "7,500 Honey" },
  { id: 12, name: "Pineapple Picking", description: "Collect 13,000 Pollen from Pineapple Patch.", objective: { type: 'pollen', field: 'pineapple', amount: 13000 }, reward: { honey: 10500 }, rewardText: "10,500 Honey" },
  { id: 13, name: "Pollen Fetcher 2", description: "Collect 16,000 Pollen.", objective: { type: 'pollen_total', amount: 16000 }, reward: { honey: 12000 }, rewardText: "12,000 Honey" },
  { id: 14, name: "Weed Wacker", description: "Collect 10,000 Pollen from various fields.", type: 'multi', tasks: [{ text: "10,000 from Strawberry", type: 'pollen', field: 'strawberry', amount: 10000 }, { text: "10,000 from Bamboo", type: 'pollen', field: 'bamboo', amount: 10000 }, { text: "10,000 from Clover", type: 'pollen', field: 'clover', amount: 10000 }], reward: { honey: 25000 }, rewardText: "25,000 Honey" },
  { id: 15, name: "Red + Blue = Gold", description: "Collect 25,000 Red and Blue Pollen.", type: 'multi', tasks: [{ text: "25,000 Blue Pollen", type: 'pollen_color', color: 'blue', amount: 25000 }, { text: "25,000 Red Pollen", type: 'pollen_color', color: 'red', amount: 25000 }], reward: { honey: 25000 }, rewardText: "25,000 Honey, 1 Gold Egg" }
];

let playerStats = { pollen_total: 0, pollen_red: 0, pollen_blue: 0, pollen_white: 0, pollen_orange: 0 };
let currentQuestIndex = 0;

function checkQuestCompletion() {
  if (currentQuestIndex >= BLACK_BEAR_QUESTS.length) return true;
  const quest = BLACK_BEAR_QUESTS[currentQuestIndex];
  let isComplete = true;
  if (quest.type === 'multi') {
      quest.tasks.forEach(task => { if (!checkSingleObjective(task)) isComplete = false; });
  } else { isComplete = checkSingleObjective(quest.objective); }
  if (questPanel.classList.contains('open')) updateQuestUI();
  return isComplete;
}
function checkSingleObjective(obj) {
    let current = 0;
    if (obj.type === 'pollen') current = playerStats[`pollen_${obj.field}`] || 0;
    else if (obj.type === 'pollen_color') current = playerStats[`pollen_${obj.color}`] || 0;
    else if (obj.type === 'pollen_total') current = playerStats.pollen_total || 0;
    return current >= obj.amount;
}
function getObjectiveProgress(obj) {
```

```
    let current = 0;
    if (obj.type === 'pollen') current = playerStats[`pollen_${obj.field}`] || 0;
    else if (obj.type === 'pollen_color') current = playerStats[`pollen_${obj.color}`] || 0;
    else if (obj.type === 'pollen_total') current = playerStats.pollen_total || 0;
    return Math.min(current, obj.amount);
}
function completeQuest() {
  if (!checkQuestCompletion()) return;
  const quest = BLACK_BEAR_QUESTS[currentQuestIndex];
  honey += (quest.reward.honey || 0);
  honeySpan.textContent = honey.toLocaleString();
  if (quest.type === 'multi') quest.tasks.forEach(task => subtractStat(task));
  else subtractStat(quest.objective);
  currentQuestIndex++;
  playerStats = { pollen_total: 0, pollen_red: 0, pollen_blue: 0, pollen_white: 0, pollen_orange: 0
};
  updateQuestUI();
}
function subtractStat(obj) {
    if (obj.type === 'pollen') playerStats[`pollen_${obj.field}`] = 0;
    if (obj.type === 'pollen_color') playerStats[`pollen_${obj.color}`] = 0;
    if (obj.type === 'pollen_total') playerStats.pollen_total = 0;
}
function updateQuestUI() {
  questListEl.innerHTML = '';
  if (currentQuestIndex >= BLACK_BEAR_QUESTS.length) {
    questListEl.innerHTML = `<div style="text-align:center; padding:20px; color:#aaffaa;">All
quests complete!</div>`;
    return;
  }
  const quest = BLACK_BEAR_QUESTS[currentQuestIndex];
  let isComplete = true;
  if (quest.type === 'multi') quest.tasks.forEach(task => { if (!checkSingleObjective(task))
isComplete = false; });
  else isComplete = checkSingleObjective(quest.objective);

  const item = document.createElement('div');
  item.className = 'quest-item';
  let progressHtml = '';
  if (quest.type === 'multi') {
      quest.tasks.forEach(task => {
          const cur = getObjectiveProgress(task);
          progressHtml += `<div class="quest-subtask">${task.text}: ${cur.toLocaleString()} /
${task.amount.toLocaleString()}</div>`;
```

```
    });
  } else {
    const cur = getObjectiveProgress(quest.objective);
    progressHtml = `<div class="quest-progress">${cur.toLocaleString()} /
${quest.objective.amount.toLocaleString()}</div>`;
  }
  item.innerHTML = `<div class="quest-name">${quest.id}. ${quest.name}</div><div
class="quest-progress">${quest.description}</div>${progressHtml}<div
class="quest-reward">Reward: ${quest.rewardText}</div><button class="quest-complete-btn"
${!isComplete ? 'disabled' : ''}>Complete Quest</button>`;
  item.querySelector('.quest-complete-btn').onclick = completeQuest;
  questListEl.appendChild(item);
}


// ========== Field data ==========
const FIELDS = {
  // 0 Bee Zone
  dandy: { name: 'Dandy Field', bonuses: { white: 0, blue: 0, red: 0, orange: 0 }, weights: {
singleWhite: 400, singleBlue: 50, singleRed: 50, doubleWhite: 50 }, totalWeight: 550,
beesRequired: 0 },
  sunflower: { name: 'Sunflower Field', bonuses: { white: 1, blue: 0, red: 0, orange: 0 }, weights: {
singleWhite: 416, doubleWhite: 46, singleBlue: 89, singleRed: 63, doubleBlue: 10, doubleRed:
10 }, totalWeight: 634, beesRequired: 0 },
  mushroom: { name: 'Mushroom Field', bonuses: { white: 0, blue: 0, red: 1, orange: 0 }, weights:
{ singleWhite: 199, singleRed: 464, doubleWhite: 22, doubleRed: 52 }, totalWeight: 737,
beesRequired: 0 },
  blueflower: { name: 'Blue Flower Field', bonuses: { white: 0, blue: 1, red: 0, orange: 0 },
weights: { singleWhite: 197, singleBlue: 461, doubleWhite: 22, doubleBlue: 51 }, totalWeight:
731, beesRequired: 0 },
  clover: { name: 'Clover Field', bonuses: { white: 0, blue: 0, red: 0, orange: 0 }, weights: {
singleWhite: 100, singleBlue: 100, singleRed: 100, doubleWhite: 100, doubleBlue: 100,
doubleRed: 100 }, totalWeight: 600, beesRequired: 0 },
  // 5 Bee Zone
  bamboo: { name: 'Bamboo Field', bonuses: { white: 0, blue: 2, red: 0, orange: 0 }, weights: {
singleBlue: 400, doubleBlue: 200, tripleBlue: 50, singleWhite: 50 }, totalWeight: 700,
beesRequired: 5 },
  spider: { name: 'Spider Field', bonuses: { white: 1, blue: 0, red: 0, orange: 0 }, weights: {
singleWhite: 73, doubleWhite: 582, tripleWhite: 73 }, totalWeight: 728, beesRequired: 5 },
  strawberry: { name: 'Strawberry Field', bonuses: { white: 0, blue: 0, red: 2, orange: 0 }, weights:
{ singleRed: 400, doubleRed: 200, tripleRed: 50, singleWhite: 50 }, totalWeight: 700,
beesRequired: 5 },
  // 10 Bee Zone
```

```
  pineapple: { name: 'Pineapple Patch', bonuses: { white: 1, blue: 0, red: 0, orange: 1 }, weights:
{ singleWhite: 200, singleOrange: 200, singleBlue: 100, doubleWhite: 100, doubleOrange: 100,
doubleBlue: 50 }, totalWeight: 750, beesRequired: 10 },
  stump: { name: 'Stump Field', bonuses: { white: 0, blue: 2, red: 0, orange: 0 }, weights: {
singleBlue: 300, doubleBlue: 200, tripleBlue: 100, singleWhite: 100, doubleWhite: 50 },
totalWeight: 750, beesRequired: 10 },
  // 15 Bee Zone
  cactus: { name: 'Cactus Field', bonuses: { white: 0, blue: 1, red: 1, orange: 0 }, weights: {
singleBlue: 150, doubleBlue: 50, singleRed: 150, doubleRed: 50, singleWhite: 50 }, totalWeight:
450, beesRequired: 15 },
  // PUMPKIN PATCH
  pumpkin: {
    name: 'Pumpkin Patch',
    bonuses: { white: 0, blue: 0, red: 0, orange: 1 },
    weights: {
       singleOrange: 50, doubleOrange: 150, tripleOrange: 300, singleWhite: 100
    },
    totalWeight: 600,
    beesRequired: 15
  },
  pine: { name: 'Pine Tree Forest', bonuses: { white: 0, blue: 2, red: 0, orange: 0 }, weights: {
singleBlue: 200, doubleBlue: 400, tripleBlue: 100, singleWhite: 50 }, totalWeight: 750,
beesRequired: 15 },
  rose: { name: 'Rose Field', bonuses: { white: 0, blue: 0, red: 2, orange: 0 }, weights: {
singleRed: 200, doubleRed: 400, tripleRed: 100, singleWhite: 50 }, totalWeight: 750,
beesRequired: 15 }
};

function getFlowerData(fieldKey) {
  const field = FIELDS[fieldKey];
  const { weights } = field;
  let totalWeight = 0;
  for (const type in weights) totalWeight += weights[type];

  const roll = Math.random() * totalWeight;
  let cumulative = 0;
  for (const type in weights) {
    cumulative += weights[type];
    if (roll < cumulative) {
      const isDouble = type.startsWith('double');
      const isTriple = type.startsWith('triple');
      const count = isTriple ? 3 : (isDouble ? 2 : 1);
      const color = type.replace(/single|double|triple/, '').toLowerCase();
      return { count, color };
```

```
    }
  }
  return { count: 1, color: 'white' };
}

// ==================== TOOLS ====================
const TOOLS = {
  hands: { name: 'Hands', range: 1, pattern: 'single', basePollen: 1, cost: 0, stats: '1 cell, 1 pollen' },
  scooper: { name: 'Scooper', range: 2, pattern: 'line', basePollen: 1, cost: 0, stats: '2 cells line, 1 pollen' },
  rake: { name: 'Rake', range: 3, pattern: 'line', basePollen: 2, cost: 500, stats: '3 cells line, 2 pollen' },
  clippers: { name: 'Clippers', range: 1, pattern: 'single', basePollen: 5, cost: 2500, stats: '1 cell, 5 pollen' },
  magnet: { name: 'Magnet', range: 3, pattern: 'surround', basePollen: 25, cost: 25000, stats: '9 cells surround, 25 pollen' },
  vacuum: { name: 'Vacuum', range: 4, pattern: 'surround', basePollen: 50, cost: 100000, stats: '16 cells surround, 50 pollen' },
  superScooper: { name: 'Super-Scooper', range: 5, pattern: 'line', basePollen: 15, cost: 40000, stats: '5 cells line, 15 pollen' },
  pulsar: { name: 'Pulsar', range: 3, pattern: 'surround', basePollen: 75, cost: 125000, stats: '9 cells surround, 75 pollen' },
  electromagnet: { name: 'Electro-Magnet', range: 2, pattern: 'surround', basePollen: 100, cost: 300000, stats: '5 cells surround, 100 pollen' },
  scissors: { name: 'Scissors', range: 6, pattern: 'line', basePollen: 25, cost: 850000, stats: '6 cells line, 25 pollen' },
  honeyDipper: { name: 'Honey Dipper', range: 4, pattern: 'surround', basePollen: 100, cost: 1500000, stats: '16 cells surround, 100 pollen' }
};

// ==================== BACKPACKS ====================
const BACKPACKS = {
  pouch: { name: 'Pouch', capacity: 200, cost: 0, stats: '200 Capacity' },
  jar: { name: 'Jar', capacity: 950, cost: 250, stats: '950 Capacity' },
  backpack: { name: 'Backpack', capacity: 3700, cost: 1000, stats: '3,700 Capacity' },
  canister: { name: 'Med. Bag', capacity: 10200, cost: 5000, stats: '10,200 Capacity' },
  megaJug: { name: 'Mega-Jug', capacity: 25000, cost: 50000, stats: '25,000 Capacity' },
  compressor: { name: 'Compressor', capacity: 50000, cost: 160000, stats: '50,000 Capacity' },
  eliteBarrel: { name: 'Elite Barrel', capacity: 125000, cost: 650000, stats: '125,000 Capacity' },
  portOHive: { name: 'Port-O-Hive', capacity: 250000, cost: 1250000, stats: '250,000 Capacity' }
};

function getCellsInRange(row, col, range, pattern) {
```

```
  const cells = [];
  if (pattern === 'single') {
    cells.push({r: row, c: col});
  } else if (pattern === 'line') {
    for (let i = 0; i < range; i++) {
      if (row + i < gridSize) cells.push({r: row + i, c: col});
    }
  } else if (pattern === 'surround') {
      const radius = Math.floor(range/2);
      for(let r = row - radius; r <= row + radius; r++){
          for(let c = col - radius; c <= col + radius; c++){
              if(r >= 0 && r < gridSize && c >= 0 && c < gridSize) cells.push({r,c});
          }
      }
  }
  return cells;
}

// ========== LOGIC & RENDER ==========

function initGame() {
    renderField();
    renderShop();
    renderInventory();
    updateUI();

    // Start Bee Loop (runs every 1 second)
    setInterval(gameLoop, 1000);

    // Initial fill
    for(let r=0; r<gridSize; r++) {
        for(let c=0; c<gridSize; c++) {
            flowerState[r][c] = { ...getFlowerData(currentField), hp: FLOWER_DURABILITY };
        }
    }
    updateFlowerVisuals();
}

function gameLoop() {
    // 1. Process Bees
    processBees();

    // 2. Refresh UI in case of changes
    updateUI();
```

```
}

function processBees() {
    if (currentBag >= bagCapacity) return;

    BEES.forEach(bee => {
        // Simple tick system: check if bee is ready
        if(!bee.lastTick) bee.lastTick = 0;
        const now = Date.now();

        if (now - bee.lastTick > bee.tickSeconds * 1000) {
            bee.lastTick = now;

            // Bee collects pollen from CURRENT FIELD
            // We'll treat this as "White" pollen for simplicity unless specific bees have color
            let amount = bee.pollenPerTick;

            // Cap at bag limit
            if(currentBag + amount > bagCapacity) amount = bagCapacity - currentBag;

            if(amount > 0) {
                currentBag += amount;

                // Bee stats count for quests!
                playerStats.pollen_total += amount;
                playerStats['pollen_' + currentField] = (playerStats['pollen_' + currentField] || 0) +
amount;
                // Defaulting bee collection to white since basic bee
                playerStats['pollen_white'] = (playerStats['pollen_white'] || 0) + amount;

                checkQuestCompletion();
            }
        }
    });
}

function renderField() {
    fieldEl.innerHTML = '';
    for(let r=0; r<gridSize; r++){
        for(let c=0; c<gridSize; c++){
            const div = document.createElement('div');
            div.className = 'flower-square';
            div.style.left = (c * cellSize) + 'px';
            div.style.top = (r * cellSize) + 'px';
```

```
            div.dataset.r = r;
            div.dataset.c = c;
            div.addEventListener('mousedown', handleFlowerClick);
            fieldEl.appendChild(div);
        }
    }
}

function updateFlowerVisuals() {
    const squares = document.getElementsByClassName('flower-square');
    for(let i=0; i<squares.length; i++) {
        const div = squares[i];
        const r = parseInt(div.dataset.r);
        const c = parseInt(div.dataset.c);
        const flower = flowerState[r][c];

        div.className = 'flower-square'; // reset
        div.innerHTML = '';

        if (flower && flower.hp > 0) {
            div.classList.add(flower.color);
            // Render petals based on count
            let content = '';
            if(flower.count === 1) content = '<div class="flower">🌸</div>';
            else if(flower.count === 2) content = '<div class="flower">🌸</div><div
class="flower">🌸</div>';
            else if(flower.count === 3) content = '<div class="flower">🌸</div><div
class="flower">🌸</div><div class="flower">🌸</div>';
            div.innerHTML = content;
        } else {
             // Cooldown visual
             div.innerHTML = '<span style="font-size:12px; opacity:0.5">🌱</span>';
        }
    }
}

function handleFlowerClick(e) {
    if (currentBag >= bagCapacity) {
        showFloatingText(e.clientX, e.clientY, "Bag Full!", "#ff5555");
        return;
    }

    const r = parseInt(e.currentTarget.dataset.r);
    const c = parseInt(e.currentTarget.dataset.c);
```

```
    const tool = TOOLS[currentTool];
    const targets = getCellsInRange(r, c, tool.range, tool.pattern);

    let totalPollen = 0;

    targets.forEach(pos => {
        const f = flowerState[pos.r][pos.c];
        if(f && f.hp > 0) {
            // Fixed: base * count only, no field bonus added to amount
            let amount = tool.basePollen * f.count;

            amount = Math.floor(amount);

            // Add to bag
            if(currentBag + amount > bagCapacity) amount = bagCapacity - currentBag;
            currentBag += amount;

            // Stats & Quests (without bonus for consistency)
            totalPollen += amount;
            playerStats.pollen_total += amount;
            playerStats['pollen_' + currentField] = (playerStats['pollen_' + currentField] || 0) +
amount;
            playerStats['pollen_' + f.color] = (playerStats['pollen_' + f.color] || 0) + amount;

            // Reduce flower HP
            f.hp--;
            if(f.hp <= 0) {
                updateFlowerVisuals(); // Update immediately to show sapling

                // Regrowth Logic
                setTimeout(() => {
                    // Respawn the flower in data
                    flowerState[pos.r][pos.c] = { ...getFlowerData(currentField), hp:
FLOWER_DURABILITY };
                    // FORCE VISUAL UPDATE
                    updateFlowerVisuals();
                }, FLOWER_RESPAWN_TIME);
            }
        }
    });

    if(totalPollen > 0) {
        checkQuestCompletion();
```

```
        showFloatingText(e.clientX, e.clientY, "+" + totalPollen, "#FFD700");
    }

    updateUI();
    updateFlowerVisuals();
}

function showFloatingText(x, y, text, color) {
    const el = document.createElement('div');
    el.className = 'pollen-popup';
    el.style.left = x + 'px';
    el.style.top = y + 'px';
    el.style.color = color;
    el.textContent = text;
    document.body.appendChild(el);
    setTimeout(() => el.remove(), 1200);
}

function updateUI() {
    honeySpan.textContent = honey.toLocaleString();
    currentBagSpan.textContent = currentBag.toLocaleString();
    maxBagSpan.textContent = bagCapacity.toLocaleString();
    fieldIndicator.textContent = FIELDS[currentField].name;

    // Bag bar color
    const pct = currentBag / bagCapacity;
    bagDisplay.style.background = `linear-gradient(90deg, #FFA500 ${pct*100}%, #fff
${pct*100}%)`;

    // Update Bees list text
    if(BEES.length > 0) {
        beesListEl.innerHTML = BEES.map(b => `<div class="bee-row"><span>${b.icon}
${b.name}</span><span>+${b.pollenPerTick}/tick</span></div>`).join('');
    } else {
        beesListEl.innerHTML = "No bees yet!";
    }
}

// ========== SHOP & INTERACTION ==========

convertBtn.onclick = () => {
    if(currentBag === 0) return;
    honey += currentBag;
    currentBag = 0;
```

```
    updateUI();
};

fieldIndicator.onclick = (e) => {
    e.stopPropagation();
    dropdown.style.display = dropdown.style.display === 'block' ? 'none' : 'block';
};

document.addEventListener('click', (e) => {
    if(!fieldIndicator.contains(e.target) && !dropdown.contains(e.target)) {
        dropdown.style.display = 'none';
    }
});

// Dropdown Items
document.querySelectorAll('.dropdown-item').forEach(item => {
    item.onclick = () => {
        const key = item.dataset.field;
        const required = FIELDS[key].beesRequired;
        if(BEES.length < required) {
            alert(`You need ${required} bees to enter ${FIELDS[key].name}!`);
            return;
        }
        currentField = key;
        dropdown.style.display = 'none';

        // Reset field
        for(let r=0; r<gridSize; r++) {
            for(let c=0; c<gridSize; c++) {
                flowerState[r][c] = { ...getFlowerData(currentField), hp: FLOWER_DURABILITY };
            }
        }
        updateFlowerVisuals();
        updateUI();
    };
});

function renderShop() {
    // Tools
    toolsList.innerHTML = '';
    for(const key in TOOLS) {
        const t = TOOLS[key];
        const div = document.createElement('div');
        div.className = `item ${toolsBought[key] ? 'bought' : ''}`;
```

```
        div.innerHTML = `
            <div class="item-name">${t.name}</div>
            <div class="item-stats">${t.stats}</div>
            <div class="item-cost">${toolsBought[key] ? 'Owned' : t.cost.toLocaleString() + '
Honey'}</div>
            <button class="buy-btn" ${toolsBought[key] ? 'disabled' : ''}>${toolsBought[key] ? 'Equip'
: 'Buy'}</button>
        `;
        const btn = div.querySelector('button');
        btn.onclick = () => {
            if(toolsBought[key]) {
                currentTool = key;
                alert(`Equipped ${t.name}`);
            } else {
                if(honey >= t.cost) {
                    honey -= t.cost;
                    toolsBought[key] = true;
                    currentTool = key;
                    renderShop();
                    updateUI();
                } else {
                    alert("Not enough honey!");
                }
            }
        };
        toolsList.appendChild(div);
    }

    // Backpacks
    backpacksList.innerHTML = '';
    for(const key in BACKPACKS) {
        const b = BACKPACKS[key];
        const div = document.createElement('div');
        div.className = `item ${backpacksBought[key] ? 'bought' : ''}`;
        div.innerHTML = `
            <div class="item-name">${b.name}</div>
            <div class="item-stats">${b.stats}</div>
            <div class="item-cost">${backpacksBought[key] ? 'Owned' : b.cost.toLocaleString() + '
Honey'}</div>
            <button class="buy-btn" ${backpacksBought[key] ? 'disabled' :
''}>${backpacksBought[key] ? 'Equip' : 'Buy'}</button>
        `;
        const btn = div.querySelector('button');
        btn.onclick = () => {
```

```
        if(backpacksBought[key]) {
            bagCapacity = b.capacity;
            alert(`Equipped ${b.name}`);
            updateUI();
        } else {
            if(honey >= b.cost) {
                honey -= b.cost;
                backpacksBought[key] = true;
                bagCapacity = b.capacity;
                renderShop();
                updateUI();
            } else {
                alert("Not enough honey!");
            }
        }
    };
    backpacksList.appendChild(div);
}

// Eggs
eggsList.innerHTML = '';
const eggDiv = document.createElement('div');
eggDiv.className = 'item';
const priceIdx = Math.min(eggsBoughtCount.basic, BASIC_EGG_PRICES.length - 1);
const price = BASIC_EGG_PRICES[priceIdx];
eggDiv.innerHTML = `
    <div class="item-name">Basic Egg</div>
    <div class="item-stats">Hatches a Basic Bee</div>
    <div class="item-cost">${price.toLocaleString()} Honey</div>
    <button class="buy-btn">Buy</button>
`;
eggDiv.querySelector('button').onclick = () => {
    if(honey >= price) {
        honey -= price;
        eggsBoughtCount.basic++;
        eggsInventory.basic++;
        renderShop();
        renderInventory();
        updateUI();
    } else {
        alert("Not enough honey!");
    }
};
eggsList.appendChild(eggDiv);
```

```
}

// Menus Toggles
shopToggle.onclick = () => shopPanel.classList.toggle('open');
questToggle.onclick = () => {
    questPanel.classList.toggle('open');
    if (questPanel.classList.contains('open')) updateQuestUI();
};
inventoryToggle.onclick = () => inventoryPanel.classList.toggle('open');
hiveBtn.onclick = () => {
    hiveView.style.display = 'block';
    renderHive();
};
closeHive.onclick = () => hiveView.style.display = 'none';

// Inventory Tabs
invTabs.forEach(tab => {
    tab.onclick = () => {
        invTabs.forEach(t => t.classList.remove('active'));
        tab.classList.add('active');
        document.querySelectorAll('.inv-content').forEach(c => c.style.display = 'none');
        document.getElementById(tab.dataset.tab + '-tab').style.display = 'block';
    };
});

function renderInventory() {
    inventoryGrid.innerHTML = '';
    if(eggsInventory.basic > 0) {
        const div = document.createElement('div');
        div.className = 'inv-item';
        div.textContent = `🥚 x${eggsInventory.basic}`;
        div.draggable = true;
        div.ondragstart = (e) => {
            e.dataTransfer.setData('type', 'basic_egg');
        };
        inventoryGrid.appendChild(div);
    }
}

function renderHive() {
    hiveGrid.innerHTML = '';
    hiveSlots.forEach((bee, index) => {
        const slot = document.createElement('div');
        slot.className = `hive-slot ${!bee ? 'empty' : ''}`;
```

```
        if(bee) {
            slot.innerHTML = `<span class="bee-icon">${bee.icon}</span>`;
        } else {
            slot.textContent = "Empty";
            slot.ondragover = (e) => e.preventDefault();
            slot.ondrop = (e) => {
                const type = e.dataTransfer.getData('type');
                if(type === 'basic_egg') {
                    if(eggsInventory.basic > 0) {
                        eggsInventory.basic--;
                        // Clone the bee stats object so every bee has its own timer
                        const newBee = { ...BEE_STATS.basic, lastTick: Date.now() };
                        hiveSlots[index] = newBee;
                        BEES.push(newBee);
                        renderInventory();
                        renderHive();
                        alert("Hatched a Basic Bee!");
                    }
                }
            };
        }
        hiveGrid.appendChild(slot);
    });
}

// Start
initGame();

</script>
</body>
</html>
```