Collision Box Diagnostic & Fix Guide

🔍 Analysis of Your Hero's Collision Box

From `EntityFactory.java`, your hero has:

```java
player.addComponent(new CollisionBox(-20, -40, 45, 95));
```

Breakdown:

- `offsetX: -20` → Box starts 20 pixels LEFT of center
- `offsetY: -40` → Box starts 40 pixels ABOVE center
- `width: 45` → Box is 45 pixels wide
- `height: 95` → Box is **95 pixels tall** (almost 1.5 tiles!)

Actual Bounds:

```
Left:   center.x - 20
Right:  center.x - 20 + 45 = center.x + 25
Top:    center.y - 40
Bottom: center.y - 40 + 95 = center.y + 55
```

⚠️ Problem Identified

Your collision box is **95 pixels tall** and **45 pixels wide**, which is:

- **1.48 tiles tall** (95 ÷ 64)
- **0.70 tiles wide** (45 ÷ 64)

This means when pathfinding thinks a tile is walkable (checking center point), the collision box actually overlaps into **neighboring tiles**!

🎯 Tile Size Reference

```
Tile size: 64x64 pixels
Hero collision box: 45x95 pixels


Visual representation:
```

```
┌───────────────┐
│  TILE 1       │ ← 64 pixels
├───────────────┤
│  TILE 2       │ ← 64 pixels
│          │
│  [HERO]       │ ← Hero spans 1.5 tiles!
│  COLLISION │
│   BOX       │
└───────────────┘
```

🛠️ Solution Options

Option 1: Reduce Collision Box (Recommended)

Make it fit within one tile:

```java
// In EntityFactory.createPlayer():
player.addComponent(new CollisionBox(-16, -24, 32, 48));
```

New dimensions:

- Width: 32 pixels (0.5 tiles) - Fits easily
- Height: 48 pixels (0.75 tiles) - Fits within 1 tile
- Still covers character body nicely

Option 2: Keep Large Box + Use Fixed Pathfinding

Keep your current 95-pixel height and use the collision-aware pathfinding I provided.

Pros:

- More realistic collision
- Better fit to sprite

Cons:

- Pathfinding is more restrictive
- May not fit through narrow passages

Option 3: Hybrid Approach

```java
// Taller but not as extreme:
player.addComponent(new CollisionBox(-20, -32, 40, 64));
```

- Width: 40 pixels
- Height: 64 pixels (exactly 1 tile)

📊 Testing Your Current Setup

Add this debug code to `Engine.java` in `keyPressed()`:

```java

```

```
// Press 'B' to check collision box
if (e.getKeyCode() == KeyEvent.VK_B) {
    Entity player = gameState.getPlayer();
    Position pos = player.getComponent(Position.class);
    CollisionBox box = player.getComponent(CollisionBox.class);

    if (pos != null && box != null) {
        float left = box.getLeft(pos.x);
        float right = box.getRight(pos.x);
        float top = box.getTop(pos.y);
        float bottom = box.getBottom(pos.y);

        int leftTile = (int)(left / 64);
        int rightTile = (int)(right / 64);
        int topTile = (int)(top / 64);
        int bottomTile = (int)(bottom / 64);

        System.out.println("\n===== COLLISION BOX DEBUG =====");
        System.out.println("Center: (" + (int)pos.x + ", " + (int)pos.y + ")");
        System.out.println("Box bounds:");
        System.out.println("  Left:   " + (int)left + " (tile " + leftTile + ")");
        System.out.println("  Right:  " + (int)right + " (tile " + rightTile + ")");
        System.out.println("  Top:    " + (int)top + " (tile " + topTile + ")");
        System.out.println("  Bottom: " + (int)bottom + " (tile " + bottomTile + ")");
        System.out.println("Spanning tiles:");
        System.out.println("  Horizontal: " + (rightTile - leftTile + 1) + " tiles");
        System.out.println("  Vertical:   " + (bottomTile - topTile + 1) + " tiles");
        System.out.println("================================\n");
    }
}
```

🔧 Recommended Implementation Steps

Step 1: Update Pathfinder.java

Replace with the collision-aware version I provided.

Step 2: Add Helper Method to GameLogic.java

```
java
```

```java
private List<int[]> findPathForEntity(Entity entity, int startX, int startY, int goalX, int goalY) {
    Pathfinder pathfinder = state.getPathfinder();
    CollisionBox collisionBox = entity.getComponent(CollisionBox.class);

    if (collisionBox != null) {
        pathfinder.setCollisionBox(collisionBox);
    }

    List<int[]> path = pathfinder.findPath(startX, startY, goalX, goalY);

    pathfinder.clearCollisionBox();

    return path;
}
```

Step 3: Update All Pathfinding Calls

Replace every instance of:

```java
state.getPathfinder().findPath(...)
```

With:

```java
findPathForEntity(entity, ...)
```

Step 4: (Optional) Adjust Collision Box

In `EntityFactory.createPlayer()`, try:

```java
// Original (too tall):
// player.addComponent(new CollisionBox(-20, -40, 45, 95));

// Recommended (fits in 1 tile):
player.addComponent(new CollisionBox(-16, -24, 32, 48));
```

🎮 Expected Results

Before Fix:

```
Click tile → Pathfinder says "walkable!"
Hero moves → Collision box hits wall
Result: STUCK (infinite loop)
```

After Fix:

```
Click tile → Pathfinder checks "Can 45x95 box fit?"
If yes: Creates valid path
If no: Finds nearby tile OR rejects path
Result: Never gets stuck!
```

📈 Performance Impact

Collision-aware pathfinding:

- Adds ~10-20 collision checks per pathfinding call
- Each check is very fast (< 0.1ms)
- Total impact: negligible (~1-2ms worst case)

When it activates:

- Only during pathfinding (not every frame)
- Player: ~1-5 times per second (clicking)
- Monsters: ~1 time per second (AI updates)

✅ Verification Checklist

After implementing:

1. ✓ Click near walls → Should path around, not into
2. ✓ Click tight corners → Should find alternative or reject
3. ✓ Debug console → Should show "✓ Found alternative path"
4. ✓ No infinite "⚠ Player stuck!" spam
5. ✓ Movement feels smooth and responsive

🐛 If Still Having Issues

Check these common mistakes:

1. **Forgot to set collision box in pathfinder**
   - Make sure `pathfinder.setCollisionBox(box)` is called
2. **Using old pathfinder.findPath() directly**
   - Must use `findPathForEntity()` wrapper
3. **Collision box offset is wrong**
   - Verify with 'B' key debug output
4. **TileMap.collidesWithTiles() not working**
   - Test by printing collision results

Your hero should now navigate perfectly!