**🔍 Analysis: Ball Jumping Between Blocks - Root Causes**

Based on my code analysis, here are the most likely causes of the ball jumping issue:

**1. 🎯 PRIMARY SUSPECT: Ground Contact Detection Logic**

The

BallTouchingTheGroundThresholdAsDotProductResult = 0.866f

 (30°) is used in **two different ways**:

**In PhysicObject.cs (Line 137):**

csharp

CopyInsert

if ((-fDot) > Definitions.BallTouchingTheGroundThresholdAsDotProductResult)

this.OnGroundContact(pCollision.gameObject, true);

**In Block.cs (Line 201):**

csharp

CopyInsert

if (s > Definitions.BallTouchingTheGroundThresholdAsDotProductResult)

this.SetContact(true);

**⚠️ ISSUE**: The PhysicObject uses (-fDot) while Block uses

s

 directly. This inconsistency could cause:

* Different contact detection between player and blocks
* Intermittent ground contact detection
* Ball "losing contact" momentarily between blocks

**2. 🔧 PHYSICS ENGINE ISSUES**

**A. Continuous Collision Detection Missing:**

* No evidence of

CollisionDetectionMode.ContinuousDynamic

* Fast-moving ball might tunnel through collision detection gaps
* Default discrete collision detection can miss contacts at block edges

**B. Rigidbody Sleep Threshold:**

csharp

CopyInsert

this.GetComponent<Rigidbody>().sleepThreshold = 0.0f;

* Sleep disabled completely - ball never "settles"
* Could cause micro-vibrations that trigger false edge detections

**3. 🎲 POTENTIAL BLOCK ALIGNMENT ISSUES**

**A. No Scale Modifications Found:**

* Blocks use default Unity cube scale (1,1,1)
* BUT: No explicit verification that prefabs are exactly 1x1x1 units
* Tiny gaps (0.001 units) would be invisible but cause physics issues

**B. Floating Point Precision:**

* Block positions from XML might have floating-point rounding errors
* Adjacent blocks at positions like (1.0000001, 0, 0) and (2.0000001, 0, 0) create micro-gaps

**4. 🎪 GRAVITY SYSTEM COMPLEXITY**

csharp

CopyInsert

this.GetComponent<Rigidbody>().AddForce(Gravity); // Every FixedUpdate

* Custom gravity applied via

AddForce

 instead of Unity's built-in gravity

* Could interact poorly with Unity's physics solver
* Might cause inconsistent force application at block boundaries

**🔍 Recommended Investigation Steps:**

**Immediate Checks:**

1. **Verify block prefab scales** - Check if Unity prefabs are exactly 1x1x1 units
2. **Add collision debugging** - Log collision normals and dot products when jumping occurs
3. **Check block positioning precision** - Verify adjacent blocks have exact integer coordinates
4. **Test with Unity's built-in gravity** - Temporarily disable custom gravity system

**Likely Quick Fixes:**

1. **Standardize dot product calculation** between PhysicObject and Block
2. **Enable continuous collision detection** on the player Rigidbody
3. **Add small overlap** between adjacent blocks (0.01 units) to eliminate micro-gaps
4. **Increase ground contact threshold** from 30° to 45° (0.707) for more forgiving detection

The **dot product inconsistency** is the most suspicious - it could cause the ball to lose ground contact detection right at block edges, causing the physics engine to apply different forces and create the jumping effect.