Ball Skill - Consistency Weighting & Anti-Siphoning System

How Consistency Protects Prize Pools

The Rating Decay System

```javascript
// Effective Rating Calculation
effectiveRating = baseRating * consistencyMultiplier * activityMultiplier

// Consistency Multiplier (based on recent performance variance)
consistencyMultiplier = Math.max(0.7, 1.2 - (standardDeviation / 200))

// Activity Multiplier (anti-dormancy protection)
daysSinceLastGame = (today - lastGameDate) / (1000 * 60 * 60 * 24)
activityMultiplier = Math.max(0.5, 1.0 - (daysSinceLastGame / 30) * 0.1)
```

Real-World Example

Player A - The Grinder (Protected)

- Base Rating: 1300
- Last 10 games: [1290, 1310, 1305, 1295, 1315, 1300, 1285, 1320, 1310, 1295]
- Standard Deviation: 12.5 (very consistent)
- Games in last 30 days: 25
- **Consistency Multiplier**: 1.14 (1.2 - 12.5/200)
- **Activity Multiplier**: 1.0 (active player)
- **Effective Rating**: 1300 × 1.14 × 1.0 = **1482**

Player B - The Dormant Sniper (Penalized)

- Base Rating: 1350 (higher than Player A)
- Last 10 games: [1200, 1500, 1100, 1600, 1250, 1450, 1180, 1520, 1300, 1400]
- Standard Deviation: 165 (very inconsistent)
- Games in last 30 days: 2 (mostly inactive)
- **Consistency Multiplier**: 0.375 (1.2 - 165/200)
- **Activity Multiplier**: 0.73 (1.0 - 28/30 * 0.1)

- **Effective Rating**: 1350 × 0.375 × 0.73 = **369**

Prize Pool Protection Mechanisms

1. Entry Fee Scaling

```javascript
// Higher-rated players pay more, but inconsistent players pay premium
entryFee = baseEntryFee * Math.pow(effectiveRating / 1000, 1.5)

// Examples:
// Effective Rating 1000: $5.00 entry
// Effective Rating 1200: $7.35 entry
// Effective Rating 1500: $13.69 entry
// Inconsistent 1500 → 600 effective: $2.78 entry
```

2. Prize Distribution Weighting

```javascript
// Prize share based on consistency-adjusted performance
prizeShare = (consistencyMultiplier * performanceScore) / totalWeightedPerformance

// This means:
// - Consistent players get higher prize percentages
// - Inconsistent players get lower prize percentages
// - Platform revenue is protected from variance exploitation
```

3. Tournament Bracket Seeding

```javascript
// Use effective rating for matchmaking, not base rating
tournamentSeed = effectiveRating + recentFormBonus

// Benefits:
// - Dormant high-rated players face appropriate competition
// - Active consistent players get fair matchups
// - No "sandbagging" with stale ratings
```

Revenue Protection Metrics

Platform Health Indicators

Monthly Active Revenue (MAR)

- Only count revenue from players with activity multiplier > 0.8
- Tracks sustainable, engaged user revenue
- Identifies and mitigates "siphon" scenarios

Consistency Ratio

- Platform average consistency multiplier
- Target: Keep above 0.9 (indicates engaged, regular players)
- Red flag: Drops below 0.8 (too many dormant high-rated players)

Prize Pool Efficiency

```javascript
efficiency = totalPrizesWonByActiveUsers / totalPrizesDistributed
// Target: >85% of prizes go to consistently active players
```

Implementation Examples

Database Schema Addition

```sql
```

```
ALTER TABLE user_ratings ADD COLUMN consistency_score DECIMAL(3,2) DEFAULT 1.00;
ALTER TABLE user_ratings ADD COLUMN activity_score DECIMAL(3,2) DEFAULT 1.00;
ALTER TABLE user_ratings ADD COLUMN effective_rating INTEGER DEFAULT 1000;

-- Trigger to recalculate on each game
CREATE OR REPLACE FUNCTION update_effective_rating()
RETURNS TRIGGER AS $$
BEGIN
    -- Calculate and update effective rating
    UPDATE user_ratings
    SET effective_rating = GREATEST(500,
        LEAST(2000,
            current_rating * consistency_score * activity_score
        )
    )
    WHERE user_id = NEW.user_id AND game_type = NEW.game_type;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

API Response Structure

json

```
{
  "userId": 12345,
  "ratings": {
    "3PT": {
      "baseRating": 1350,
      "effectiveRating": 1482,
      "consistencyScore": 1.14,
      "activityScore": 1.0,
      "gamesPlayed": 87,
      "recentForm": "excellent"
    }
  }
}
```

Business Benefits

1. Sustainable Economics

- Prevents prize pool drain from dormant players
- Ensures active players get fair reward distribution
- Creates incentive for consistent platform engagement

2. Competitive Integrity

- Eliminates "rating camping" strategies
- Encourages regular play to maintain standing
- Fair matchmaking based on current skill level

3. User Retention

- Rewards consistent players with better prizes
- Creates daily/weekly engagement loops
- Builds community of active, invested players

4. Revenue Optimization

- Entry fees scale appropriately with true skill
- Reduces payout variance risk
- Attracts serious, committed players

Adjustment Parameters

Tunable Constants

```javascript
const CONSISTENCY_WEIGHT = 0.4;  // How much consistency matters (0-1)
const ACTIVITY_DECAY_RATE = 0.1; // How fast dormancy penalty kicks in
const MIN_EFFECTIVE_RATING = 500; // Floor to prevent over-penalization
const MAX_EFFECTIVE_RATING = 2000; // Ceiling for system stability
```

A/B Testing Opportunities

- Test different decay rates for optimal engagement
- Experiment with consistency weight for prize distribution
- Optimize entry fee scaling for revenue vs. participation

This system ensures your most valuable users (consistent, active players) get the best experience and largest prize shares, while preventing exploitation of the prize pool by dormant players with inflated

ratings.