Release Candidate Testing: The final build before release underwent rigorous testing to ensure it met all functional and performance requirements and was free of critical bugs.

Player Feedback: Feedback was actively sought through dedicated Discord channels and structured surveys. If players found the DDA too punishing or too lenient, its parameters were adjusted.

Bug Tracking: Bugs were systematically tracked using GitHub Issues, categorized by severity, and addressed in regular bug-fix sprints.

Research Design of DDA

This study employs a quantitative experimental design to evaluate a rule-based Dynamic Difficulty Adjustment (DDA) system implemented in a poker-based roguelike game. The research methodology combines software development, system modeling, and empirical testing to assess the effectiveness of performance-based difficulty scaling in maintaining player engagement and achieving flow state.

3.1.1 System Development Approach

The DDA system was developed using TypeScript and follows a singleton design pattern to ensure consistent state management throughout gameplay sessions. The implementation draws inspiration from established roguelike titles including Hades, Dead Cells, and Risk of Rain 2, which employ adaptive difficulty mechanisms based on player performance rather than binary win/loss outcomes.

3.2 DDA System Architecture

3.2.1 Core Components

The DDA system consists of four primary components:

- **Performance Tracking Module**: Captures combat metrics including health retention, turn efficiency, hand quality, damage output, and resource management
- **Player Performance Score (PPS) Calculator**: Processes combat metrics to generate a continuous skill rating between 0.0 and 5.0
- **Difficulty Tier Manager**: Maps PPS values to four difficulty tiers (Struggling, Learning, Thriving, Mastering)
- **Adjustment Application Layer**: Modifies game parameters based on current difficulty tier

3.2.2 Player Performance Score (PPS) System

The PPS serves as the central metric for difficulty assessment, operating on a continuous scale from **0.0 (minimum skill)** to **5.0 (maximum skill)**. The system tracks performance across multiple dimensions. The PPS is updated after each combat encounter using the following core formula, which ensures the score is always clamped between 0 and 5:

$$PPSnew \ = \ max\,(0, min(5, PPScurrent \ + \ \Delta PPS))$$

Where PPScurrent is the current score and ΔPPS is the calculated adjustment based on the sum of eight performance factors (F1 through F8).

$$\Delta PPS = \sum_{i=1}^{8} Fi$$

The calculation of ΔPPS incorporates eight performance factors, summarized as follows:

Performance Metrics

- **Health Retention Performance**
  - Excellent retention (90-100% HP): +0.35 base adjustment
  - Good retention (70-89% HP): +0.15 base adjustment
  - Moderate retention (50-69% HP): 0.0 adjustment
  - Poor retention (30-49% HP): -0.20 base adjustment
  - Critical retention (<30% HP): -0.40 base adjustment
  - Perfect combat (no damage): additional +0.25 bonus
- **Skill Expression Metrics**
  - Excellent hands (**Four of a Kind or better**): +0.25 base adjustment
  - Good hands (**Straight or better**): +0.10 base adjustment
  - Resource efficiency (≤30% discard usage): +0.15 base adjustment
- **Combat Efficiency Performance**
  - Turn count is evaluated against tier-specific expectations
  - Efficient completion: +0.20 base adjustment

    - Inefficient completion: -0.20 base adjustment
- **Damage Efficiency Performance**
    - High damage-per-turn ratio (≥130% of expected): +0.20 base adjustment
    - Low damage-per-turn ratio (≤70% of expected): -0.15 base adjustment

3.2.3 Difficulty Tier System

Four distinct difficulty tiers partition the PPS continuum:

Tier	PPS Range	Description
Struggling	0.0 - 1.0	Player requires assistance
Learning	1.1 - 2.5	Standard balanced difficulty
Thriving	2.6 - 4.0	Player demonstrating mastery
Mastering	4.1 - 5.0	Expert-level challenge

3.2.4 Tier-Based Modifier Scaling

To prevent player stagnation and encourage progression, the system implements asymmetric scaling of performance adjustments based on current tier:

Struggling Tier Scaling

- Penalty multiplier: 0.5× (reduced penalties)
- Bonus multiplier: 1.5× (amplified rewards)
- Design rationale: Accelerate recovery from poor performance

Learning Tier Scaling

- Penalty multiplier: 1.0× (standard)
- Bonus multiplier: 1.0× (standard)
- Design rationale: Neutral, balanced assessment

Thriving Tier Scaling

- Penalty multiplier: 1.2× (increased penalties)
- Bonus multiplier: 0.8× (reduced rewards)
- Design rationale: Maintain challenge for skilled players

Mastering Tier Scaling

- Penalty multiplier: 1.5× (maximum penalties)
- Bonus multiplier: 0.5× (minimal rewards)
- Design rationale: Preserve elite-level difficulty

3.2.5 Advanced Adjustment Systems

Calibration Period The system implements an initial calibration phase spanning the first three combat encounters. During calibration:

- PPS adjustments are tracked but difficulty tier remains fixed at "Learning"
- Prevents premature difficulty spikes or drops
- Allows system to establish baseline player skill level

Comeback Momentum System Activated when PPS falls below 1.5 (deep in Struggling tier):

- Base momentum bonus: +0.30 per positive performance
- Consecutive victory bonus: +0.15 per consecutive good performance (capped at +0.45)
- Design rationale: Prevent downward spirals and encourage recovery

Clutch Performance Recognition Awards bonuses for strong performance when entering combat with low health (<50%):

- Bonus scales with initial disadvantage (0.0 to 1.0 multiplier)
- Maintains HP percentage: +0.20 × disadvantage × tier multiplier
- Victory despite disadvantage: +0.15 × disadvantage × tier multiplier
- Design rationale: Reward skillful play under pressure

3.3 Difficulty Adjustment Parameters

3.3.1 Enemy Scaling

Enemy parameters adjust dynamically based on current difficulty tier:

Tier	Health Multiplier	Damage Multiplier	AI Complexity
Struggling	0.8×	0.8×	0.5
Learning	1.0×	1.0×	1.0
Thriving	1.15×	1.15×	1.3
Mastering	1.15×	1.15×	1.5

3.3.2 Economic Adjustments

Shop prices and gold rewards scale inversely to provide appropriate resource availability:

Tier	Shop Price Multiplier	Gold Reward Multiplier
Struggling	0.8× (-20%)	1.2× (+20%)
Learning	1.0× (baseline)	1.0× (baseline)
Thriving	1.1× (+10%)	0.9× (-10%)
Mastering	1.2× (+20%)	0.8× (-20%)

3.3.3 Map Generation Bias

Node distribution probabilities adjust to influence encounter frequency:

Tier	Rest Nodes	Combat Nodes	Elite Nodes
Struggling	30%	40%	10%
Learning	20%	50%	15%
Thriving	15%	50%	20%
Mastering	10%	40%	25%

3.4 Alternative Configurations

To facilitate comparative analysis, three distinct configuration profiles were developed:

3.4.1 Default Configuration

Balanced approach with moderate adjustment rates and 3-combat calibration period. Designed for broad player appeal and stable difficulty progression.

3.4.2 Aggressive Configuration

- No calibration period (immediate adjustments)
- Increased adjustment magnitudes (up to 67% larger)
- Reduced assistance in Struggling tier
- Disabled comeback momentum system
- Target audience: Experienced roguelike players seeking challenge

3.4.3 Conservative Configuration

- Extended 5-combat calibration period
- Reduced penalty severity (up to 70% reduction in Struggling tier)
- Enhanced comeback momentum (+67% stronger bonuses)
- Doubled bonuses for Struggling tier players
- Target audience: Players new to roguelike genre

3.5 Data Collection and Metrics

3.5.1 Session-Level Metrics

The system captures comprehensive session data:

- Total combat encounters completed
- Victory and defeat counts
- Average health retention per combat
- Average turn count per combat
- Hand quality distribution
- PPS trajectory over time
- Difficulty tier transitions

3.5.2 Combat-Level Metrics

Each combat encounter records:

- Pre-combat state (health, gold, resources)

- Performance metrics (damage dealt/received, turns, hands played)
- Post-combat state and outcomes
- PPS adjustment magnitude and contributing factors

3.5.3 Event Logging

All DDA-related events are timestamped and logged:

- PPS updates with adjustment breakdowns
- Tier transitions with triggering factors
- Configuration changes
- Calibration phase completion
- System enable/disable toggles

3.6 Testing Methodology

3.6.1 Unit Testing Framework

A comprehensive test suite of 47 unit tests validates DDA system behavior across scenarios:

Tier-Specific Scaling Tests (12 tests)

- Verify correct multiplier application for each tier
- Validate boundary conditions between tiers
- Confirm expected turn thresholds per tier

Clutch Performance Tests (4 tests)

- Validate bonus calculation at various starting health levels
- Confirm activation thresholds
- Test bonus scaling with disadvantage

Resource Management Tests (2 tests)

- Verify efficiency bonus activation
- Confirm threshold-based reward system

Multi-Factor Interaction Tests (6 tests)

- Test combined positive factors
- Test combined negative factors
- Test mixed positive/negative scenarios
- Validate tier scaling across all factors

Edge Case and Boundary Tests (8 tests)

- Test exact boundary values (70%, 90% health)
- Validate tier transition points
- Test with varying enemy health pools
- Verify PPS clamping (0.0-5.0 range)

Basic Functionality Tests (12 tests)

- Core bonus/penalty application
- Calibration period behavior
- PPS clamping

- Tier transition logic

All expected values in tests are manually calculated from game design specifications to ensure accuracy.

3.6.2 Validation Approach

Tests employ a "set and verify" methodology:

1. Initialize DDA system with known state
2. Process combat scenario with specific metrics
3. Calculate expected PPS adjustment manually
4. Compare system output to expected values (±0.02 tolerance)
5. Verify tier assignment matches PPS thresholds

3.7 Implementation Details

3.7.1 State Management

The system maintains persistent state across combat encounters:

- Current and previous PPS values
- Combat history (rolling 10-encounter window)
- Consecutive victory/defeat counters
- Calibration status
- Session start time and total combats completed

3.7.2 Performance Optimization

To ensure real-time responsiveness:

- Combat history limited to 10 most recent encounters
- Event log capped at 100 entries
- Singleton pattern prevents multiple instance overhead
- State snapshots enable backup/restore functionality

3.7.3 Debugging and Transparency

Console logging provides real-time insight into DDA decisions:

- Detailed PPS calculation breakdowns
- Tier assignment rationale
- Contributing factors for each adjustment
- Current vs. expected performance metrics

3.8 Limitations and Considerations

3.8.1 System Limitations

- PPS calculation assumes consistent player skill; does not account for learning curves within sessions
- Difficulty adjustments apply globally; no per-encounter fine-tuning
- System requires minimum combat data (calibration period) before full activation
- Economic and map generation adjustments may lag behind rapid skill changes

3.8.2 Ethical Considerations

- Player awareness: System operates transparently with narrative feedback
- No manipulation: Adjustments aim to match challenge to skill, not exploit player psychology
- Optional system: Can be disabled if players prefer static difficulty
- Fair progression: All players can reach any tier through demonstrated performance

3.8.3 Technical Constraints

- Single-player focus; no multi-player considerations
- State persistence requires proper save/load implementation
- Browser-based implementation (no server-side validation)
- TypeScript implementation limits cross-platform portability

3.9 Expected Outcomes

Based on the methodology, this research anticipates:

1. **Engagement Maintenance**: PPS-based adjustment should maintain appropriate challenge levels across skill ranges
2. **Flow State Achievement**: Tier-specific scaling should balance challenge and skill according to flow theory
3. **Recovery Facilitation**: Comeback momentum system should prevent player dropout during difficulty spikes
4. **Skill Recognition**: System should accurately identify and respond to performance improvements/declines

The methodology provides a foundation for empirical validation of DDA effectiveness in roguelike game design, with potential applications to other procedurally-generated game genres.

SUMMARY

In summary, the Game Development Life Cycle provided a robust and structured framework for the development of "Bathala." Each phase—conceptualization, pre-production, production, post-production, and maintenance—played a critical and interconnected role in shaping the final product. The process, driven by continuous feedback and iterative testing, enabled constant improvements and adaptation to design challenges, ultimately ensuring the game was engaging, well-optimized for the web, and deeply rooted in Filipino culture.

The development process involved the meticulous implementation of specific algorithms, most notably the rule-based Dynamic Difficulty Adjustment (DDA) system to maintain player flow and Procedural Generation algorithms to create unique, replayable overworlds. A Finite State Machine (FSM) was used for robust game state management, while the Fisher-Yates shuffle ensured fair card distribution. The selection of Phaser.js as the primary game framework, alongside TypeScript, proved instrumental in achieving an efficient and maintainable web-based game.

Reflecting on the overall process, the chosen GDLC proved highly effective. Challenges related to web performance optimization, game balancing, and authentic cultural integration were proactively addressed through the structured phases and iterative testing. For future projects, further integration of automated testing for complex game logic could enhance efficiency. The lessons learned from the challenges and solutions encountered throughout the GDLC will contribute valuable experience to future game development endeavors, particularly in combining cultural narratives with innovative gameplay mechanics.