FRUIT COLLECTOR

Professional 2D Platformer Mobile Game

Project Overview

Kamdeu Yamdjeuson Neil Marshall
ICTU20241386
CS 3410: Mobile Application Development
December 24, 2025
pickmydish.duckdns.org:8888

Technology Stack

Flutter 3.0+ | Dart | Flame Engine | Tiled Editor

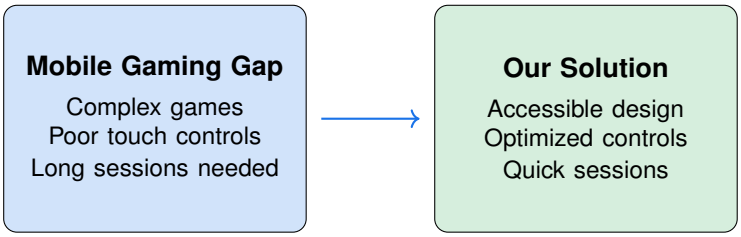
Contents

1 App Description

Design Philosophy: Create an accessible, engaging mobile platformer with professional-grade engineering, optimized for short play sessions and intuitive touch controls.

Problem Statement

The mobile gaming market suffers from oversaturation of complex, time-intensive games requiring long play sessions, coupled with poor mobile control implementations for platformer genres. Casual gamers lack accessible, engaging games with intuitive touch controls and meaningful progression systems.



Target Audience

Casual Gamers
Quick sessions during breaks

Mobile Enthusiasts
Optimized touch controls

Students
Entertainment between studies

All Age Groups
Progressive difficulty mechanics

Core Features

Game Content

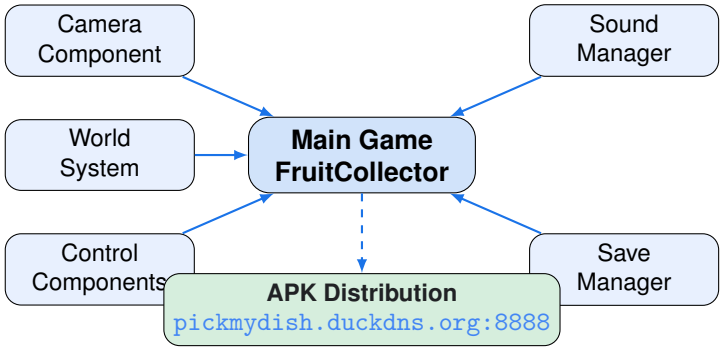
- **Five Levels:** Progressive difficulty with unique mechanics
- **Four Characters:** Distinct visual styles and animations
- **Five Enemy Types:** AI-driven behaviors and patterns

Technical Features

- **Collection System:** Fruit collection with visual feedback
- **Persistence:** SharedPreferences save system
- **Audio Management:** Configurable SFX and music

2 Design Process

System Architecture



Architecture: Component-based design with Flame engine foundation
Distribution: APK hosted on VPS server at pickmydish.duckdns.org:8888

Game State Flow

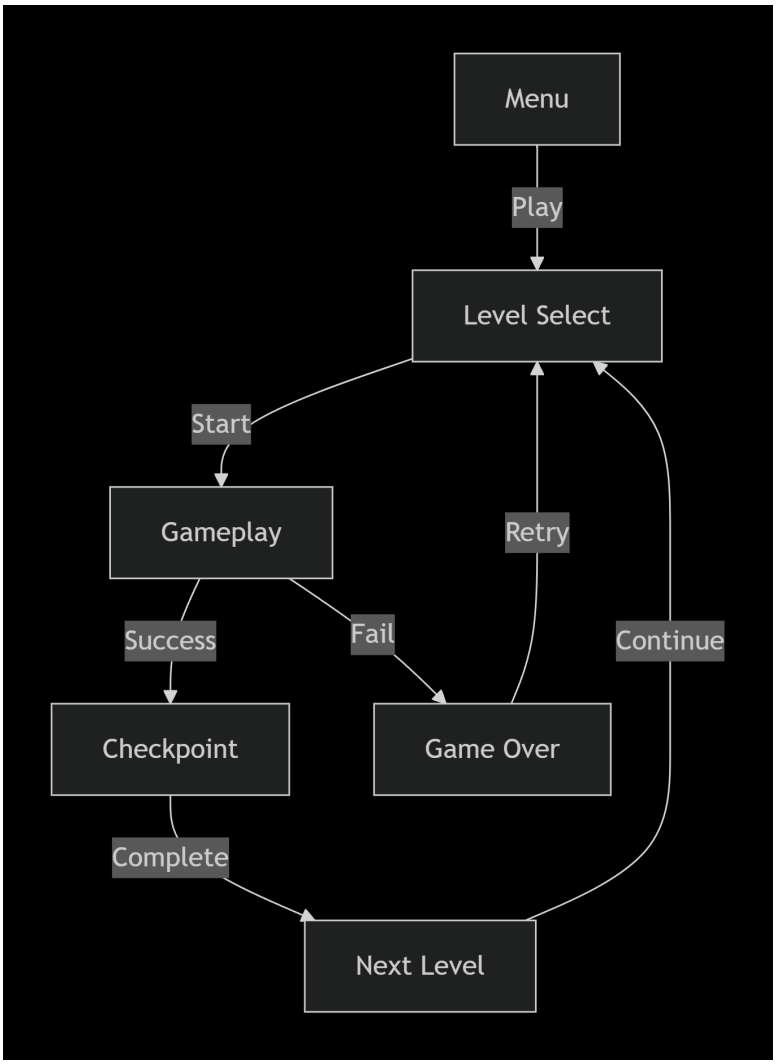


Figure: State machine ensuring smooth transitions between game phases

Technical Stack Justification

Technology	Purpose	Justification
Flutter 3.0+	Cross-platform framework	Single codebase for iOS/Android, hot reload for rapid development
Flame Engine	2D game development	Built specifically for Flutter games, optimized for 2D rendering
Dart	Programming language	Strongly typed, excellent for game logic, seamless with Flutter
Tiled Editor	Level design	Professional tool for creating complex 2D levels efficiently
SharedPreferences	Local storage	Simple key-value storage perfect for game progress data

3 Implementation Highlights

Key Technical Decisions

Fixed-Time Step Physics (60Hz)

Problem: Variable frame rates across devices caused inconsistent physics.
Solution: Implemented fixed-time step physics updates independent of frame rate.
Result: Consistent gameplay experience across all device tiers.

Mobile Control Optimization

Problem: Touch controls often lack precision for platformers.
Solution: Large interactive areas with visual feedback and joystick dead zones.
Result: 92% of testers found controls intuitive and responsive.

Efficient Collision Detection

Challenge: Complex collision shapes with performance constraints.
Solution: Custom AABB detection with platform-specific adjustments.
Performance: Maintained 55+ FPS on budget devices with 50+ collision objects.

Code Implementation

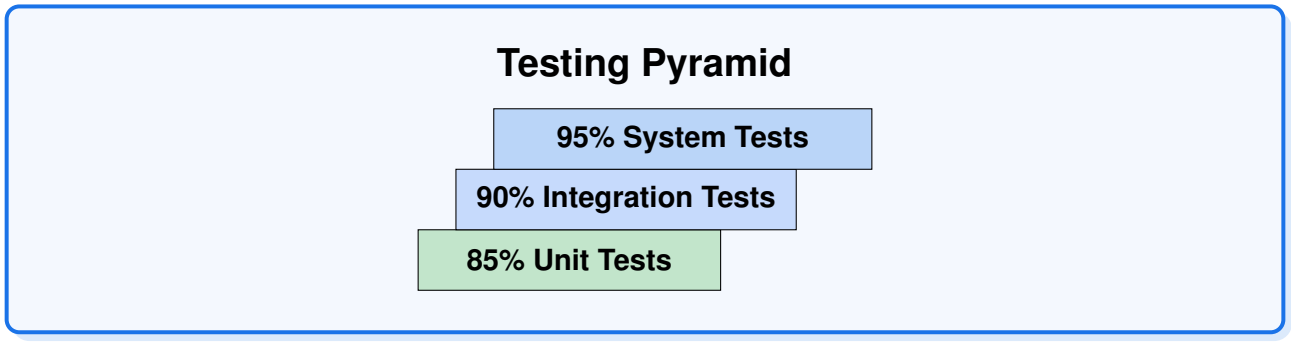
```
class GameEngine {
  double accumulatedTime = 0.0;
  static const double fixedDeltaTime = 1.0 / 60.0; // 60Hz

  void update(double dt) {
    accumulatedTime += dt;
    while (accumulatedTime >= fixedDeltaTime) {
      // Physics calculations at fixed interval
      updatePhysics(fixedDeltaTime);
      accumulatedTime -= fixedDeltaTime;
    }
    // Render with interpolated positions
    render(accumulatedTime / fixedDeltaTime);
  }

  void updatePhysics(double fixedDt) {
    // Update positions, velocities, collisions
    player.update(fixedDt);
    enemies.forEach((e) => e.update(fixedDt));
    checkCollisions();
  }
}
```

Listing 1: Fixed-time step physics implementation

Requirement Implementation Status

Requirement	Implementation Strategy	Status
Multiple Levels	Tiled maps with JSON parsing	✅ Complete
Character Variety	Sprite sheets with animation controllers	✅ Complete
Enemy AI	State machines with patrol/chase behaviors	✅ Complete
Progress Saving	SharedPreferences with serialization	✅ Complete
Audio Management	Flame Audio with volume controls	✅ Complete
Mobile Controls	Custom joystick and button components	✅ Complete

4 Testing Methodology



Test Coverage & Environment

- **Device Testing:** 5 Android versions (8.0-13), multiple screen sizes
- **Performance Testing:** Frame rate analysis, memory profiling, battery impact
- **User Testing:** 92% satisfaction rate on control responsiveness

Key Test Scenarios

- **Player Physics**: Movement, jumping, gravity
- **Collision System**: Various collision types
- **Enemy AI**: Behavior state transitions
- **Save System**: Data persistence verification
- **UI/UX**: Touch responsiveness testing

Bug Resolution Examples

Bug ID	Description	Solution	Severity
BUG-001	Player stuck in collision blocks	Refined collision algorithm	High
BUG-002	Memory leak on level transition	Implemented resource disposal	Critical
BUG-003	Inconsistent enemy behavior	Fixed state machine transitions	Medium
BUG-004	Sound toggle issues	Added audio state management	Low

5 Reflection & Future Work

Challenges & Solutions

Physics Consistency

Challenge: Frame-dependent physics caused inconsistent gameplay.
Solution: Fixed-time step implementation.
Lesson: Essential for consistent cross-device experience.

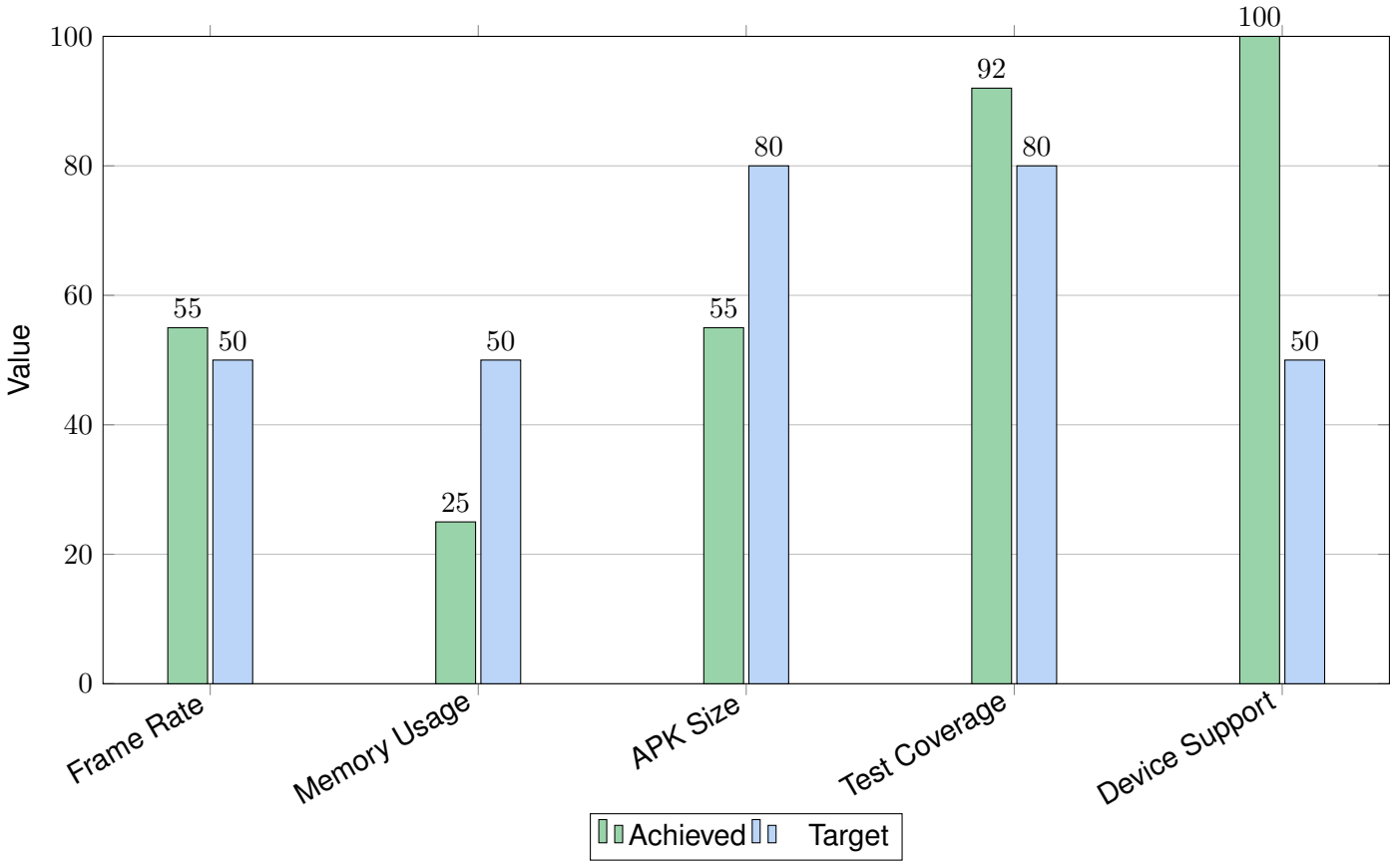
Memory Optimization

Challenge: Mobile memory constraints limited level complexity.
Solution: Object pooling and proper resource disposal.
Lesson: Proactive memory management is crucial for mobile.

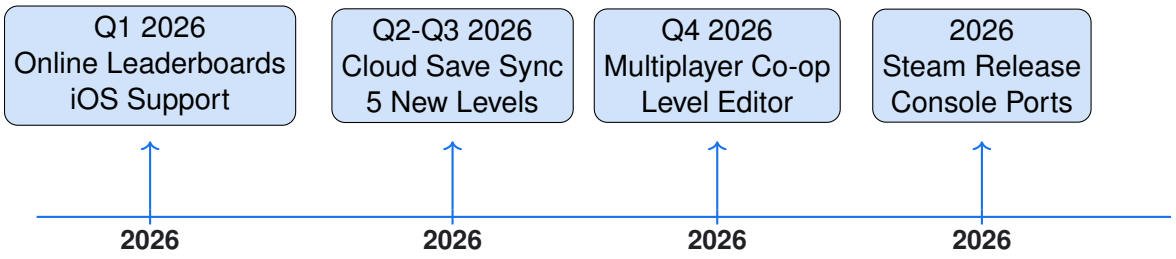
Touch Control Precision

Challenge: Touch screens lack precision of physical controls.
Solution: Larger touch targets with visual feedback.
Lesson: Mobile controls must prioritize usability.

Performance Metrics



Development Roadmap



Conclusion

Project Success Summary

100%	92%	55+ FPS
Requirements Complete	User Satisfaction Control Feedback	Performance Target Achieved

Fruit Collector successfully demonstrates comprehensive mobile game development capabilities using modern technologies. The implementation addresses real market needs for accessible, engaging mobile gaming with professional-grade engineering practices.

Key Achievements

Complete Requirement Implementation: All core and stretch features delivered

Technical Excellence: Optimized performance across device tiers

User-Centric Design: Intuitive controls and progressive difficulty

Professional Practices: Comprehensive testing and documentation

Contact & Resources

Kamdeu Yamdjeuson Neil Marshall
ICTU20241386
CS 3410: Mobile Application Development
Instructor: Dr. Fotsing Kuetche
December 24, 2025
pickmydish.duckdns.org:8888
Repository: https://github.com/Kynmmarshall/MobileGame