Reflective Development Journal

Brona Keevers-Roux

20098883

Into the Orchards

Mobile Game Dev

AI-Assisted Coding Statement

I didn't use AI for any of my code. Everything was done by myself.

Timeline of decisions, scope and risks

The development of *Into the Orchards* began with a broad concept: an idle clicker game where players collect fruit, create smoothies, and upgrade their orchards. Early notes outlined features such as multiple fruit types, automated harvesting, blender upgrades, offline resource generation, and a prestige system.

As development progressed the plan for the completed game became more solid with specific details worked out.

The UI has been designed with phone screen safe areas in mind, avoiding the camera area. The size of all the elements have been carefully decided with constant referencing to things on my own phone in order to get the scale correct.

Some cuts were made such as the game still being silent with no music or SFX due to time constraints. The blender system could also still do with more upgrading to keep up with the fruit generation.

Technical Deep-Dives

ScriptableObjects and Prefab Management

To manage game data efficiently, I used ScriptableObjects for trees, recipes, ingredients, and upgrades. This allows for modular content expansion, making it easy to add new fruits, recipes, and orchard upgrades without changing core game scripts.

Prefabs were instantiated dynamically during game load, with event-driven updates controlling UI changes. This design drastically improved framerates and avoided expensive Canvas rebuilds. Doing this, GC allocations per frame remained mostly 0, with only occasional 2–4 allocations per frame, even under peak interaction.

Learning: This approach highlighted the importance of data-driven architecture. It showed how careful design can simplify feature addition, minimize bugs, and maintain performance on mobile devices.

Offline Resource Generation System

The offline resource system was designed to reward players based on time away from the game. The initial implementation allowed multiple triggers if the player repeatedly returned to the main menu and resumed, leading to inflated resources.

The solution involved creating bool, hasCollectedOfflineResources, which prevents repeated rewards per session. The bool is reset at the beginning of each session ensuring the players aren't being prevented from receiving the offline rewards when they should be.

Learning: I learned the importance of clear session boundaries and careful testing of state persistence.

Performance Story

Performance analysis was conducted using the Unity Profiler connected to the Android Emulator. While emulation does not perfectly replicate mobile hardware, it provided a reliable baseline for frame timing, memory usage, and draw-call optimization.

Baseline Metrics

- Average frame time: ~4 ms
- CPU frame time: ~2 ms
- GPU frame time: ~2 ms
- 99th percentile: 5 ms
- GC allocations/frame: mostly 0, occasionally 2–4
- Batches: 28
- Cold start: 9 seconds
- APK size: 40.4 MB

Optimization Measures

- Event-driven UI: Canvas updates only trigger when the player interacts or a smoothie finishes crafting. This minimizes CPU usage and prevents unnecessary allocations.
- Performance-focused scope cuts: Limiting simultaneous active blenders and automated harvesters ensured CPU/GPU usage remained low.

Outcome

Post-optimization metrics show stable 60 FPS emulation with minimal GC overhead and low draw calls, demonstrating efficient rendering and responsive gameplay. Event-driven design and pooling collectively improved frame stability and memory efficiency.

Reflection: Even with emulator limitations, careful measurement and targeted optimizations gave insight into mobile performance. I learned that profiling early and iterating on design prevents major late-stage bottlenecks.

New Systems

- **Added 3 more types of fruits.**
  This gives more variation to the game allowing the player to use more valuable fruits in their smoothies.

- **Created more art.**
  Art for the additional fruits and for the padlock was created.

- **Added more orchard upgrades.**
  Fruit quality was implemented, increasing the value of smoothies crafted with the upgraded fruits. Harvester upgrades were also added for automatic resource generation.

- **Added automatic resource generation.**
  Harvesters will automatically generate fruits for the player with speed upgrades to make them gather more frequently.

- **Added offline resource generation.**
  Offline resource generation is calculated based on the players harvesters. There will be no offline resource generation if the player has no harvesters.

- **Orchard locking**
  Orchards now need to be unlocked using coins, this helps to create a more enjoyable gameplay loop with the player feeling rewarded when they get enough gold to unlock a new fruit type.

- **Added prestige system with 3 prestige upgrades.**

  Blender speeds, orchard upgrade discounts and max offline time can be upgraded with prestige points. Prestige points are earned by resetting all game progress. Prestige upgrades are permanent and don't get reset in future prestiges.

Playtesting

Playtesting was conducted using a web build hosted on itch.io, in addition to the Android Emulator and Unity Simulator, as I did not have access to a physical mobile device during this CA. Eleven participants played the game through the web build, providing feedback on controls, UI clarity, and overall engagement. Observations showed that the core gameplay loop (collecting fruit, crafting smoothies, and upgrading orchards) was generally well understood and enjoyable. Players highlighted that the controls were responsive and the UI readable, though some instructions were occasionally unclear, and suggestions were made for additional guidance and blender upgrades. The blenders automatic crafting system was difficult for several players to understand what was happening. Insights from this web-based playtesting informed targeted improvements, including bug fixes, recipe persistence, and UI adjustments. Using the itch.io build allowed me to gather real user interaction data quickly and efficiently, demonstrating the value of early external playtesting in identifying usability issues and validating the core loop before final deployment.

For future development I would focus on creating instructions and info for the player to allow them to better understand what each upgrade does and how the systems work.

References

I used Bayat Games SaveGameFree package from the unity asset store to convert my serializable data into json format for persistence.