CMPT 276 Class Project: Code Review Report
Team: 06

Game title: Arcade Game
Goal: Collect the rewards, then reach the exit. Don't get caught, and don't let your score drop below zero.

This code review was done after Phase 3 was finished. The goal was to find code smells in the code we wrote, fix them to improve the design and code quality, and still keep the program working the same way.

I (Besmillah Sultani) refactored three files in the generators area: BoardGenerator, Spawner, and SpawnerHelper. The main issues were randomness that changes every run, repeated code, confusing helper code that was left behind, too many parameters in one method, repeated validation checks, and slow performance from repeated scanning and creating objects.

In BoardGenerator, I fixed four issues. First, the class used Random in a way that was not repeatable, so tests could fail or pass depending on luck. I changed it so a Random can be passed in, and I added a seeded option for tests. Second, there was a barrier helper list that looked like normal generation logic, but it was not part of the normal flow. I kept it for compatibility, but I made its purpose clear so it does not confuse future readers. Third, the logic for picking the start and exit was copied in several places. I moved that logic into one shared workflow so all modes reuse the same steps. Fourth, the start and exit selection was mixed into the generator logic. I separated the selection rules so they can be changed or reused without rewriting the generator.

In Spawner, I fixed problems that affected correctness and readability. First, it created its own Random, which made spawning results change every run. I changed it so a Random can be passed in and seeded for tests. Second, the bonus quota logic could trigger extra bonus waves when bonuses expired or were ignored. I fixed this by reducing the remaining quota when bonuses spawn, so the total number of bonus spawns is limited. Third, one method had a long list of primitive parameters, which was hard to read and easy to mix up. I created a BonusConfig object and added an overload that takes it, so the call is clearer. In the same area, I improved performance by only choosing the first K random positions when that is all we need. Fourth, the same reachability check logic was repeated for both punishment and enemy placement. I moved this into one shared reachability check so the same rules are always used. Finally, the class was doing too many different jobs in one large section of code. I split the logic into smaller focused parts, but I kept the same public API so other code still works. I also renamed rng to random to make the meaning clear.

In SpawnerHelper, the biggest issue was performance. It repeatedly scanned the board and created many Position objects that were not needed. The BFS reachability checks also did extra work and created extra objects. I updated freeFloorCells so it only creates Position objects for cells that are actually free. I also improved the reachability checks to reduce repeated work and object creation.

After every refactoring step, I reran tests to make sure the program still behaved the same. I made the changes in small steps, and I committed and pushed often so each commit shows one clear improvement.