**Technical Decisions**

The PillAttack script is designed to handle the pill’s defense and health mechanics, as well as managing projectile firing at enemies. Several important design choices were made to keep the system modular and scalable

**Modular Enemy and Pill Interaction:**

The system uses triggers for enemy interactions(AddEnemyToStopline and RemoveEnemy)to manage when enemies reach the stop line, which directly impacts the pill's health. This approach allows flexibility in handling different numbers and types of enemies.

**Projectile Firing Mechanism:**

Projectiles are instantiated and directed toward the closest enemy. The Shoot() function handles projectile spawning and aims toward the target, with velocity set dynamically to ensure the projectile hits the target.

**Health Management:**

Pill health is reduced periodically based on the number of enemies at the stop line, with damage calculated dynamically. The health is updated via a TextMeshProUGUI component for UI feedback.

**Game Over Mechanism:**

When the pill’s health reaches zero, the game transitions to a game-over state. A GameOver panel is activated to signal the end of the game.

**Challenges Faced and Solutions Implemented**

**Managing Enemy Count at Stop Line:** The key challenge was managing the number of enemies at the stop line. I solved this by tracking the number of enemies with simple counters (enemiesAtStopline). Each enemy added or removed updates the pill’s health, making the system dynamic and responsive to gameplay changes.

**Projectile Accuracy:** Ensuring that projectiles accurately hit the nearest enemy required calculating directions and using physics (with Rigidbody2D velocity) to propel the projectiles. To improve the accuracy of projectiles, I used Mathf.Atan2 to rotate the projectiles toward the enemy and fine-tuned their velocities for appropriate speed.

**Health Reduction Over Time:** Reducing the pill's health based on enemy presence was initially tricky, as it needed to be handled over time. I implemented a timer (healthReductionTimer) to decrease health once per second, ensuring that the health reduction logic only triggers at set intervals and doesn't cause instant damage spikes.

**Wave System and Enemy Spawning:** Wave Logic Complexity: One of the major challenges was implementing a smooth and scalable wave system. Each wave has different spawn intervals and durations, which meant the logic for tracking wave progression, spawning enemies at the right times, and transitioning between waves had to be precise. I tackled this by using waveIntervals and waveDurations arrays to define spawn behavior and ensured that the transition from one wave to the next happened smoothly.

**Handling Dynamic Spawn Intervals:** The spawn intervals needed to be adjusted for each wave, and ensuring the enemies didn’t spawn too quickly or slowly was critical. I used a countdown timer (spawnTimer) to control the spawn rate and synchronized it with the wave's duration, allowing for more frequent spawns as the difficulty increased.

**Wave Transition Issues:** There were difficulties in making sure the game properly transitioned from one wave to another while keeping track of the active wave. I implemented a delay (waveStartDelay) before the next wave begins, giving players time to prepare. This delay also ensured that the game doesn’t immediately launch the next wave before the previous wave ends, which could overwhelm the system.

**Game Over Transition:** Triggering a game-over state when the pill's health reached zero was straightforward but required careful handling of UI visibility (gameOverPanel.SetActive(true)), ensuring the game paused at the correct moment and displayed the game-over screen properly.

**What I Would Improve With More Time**

**Enemy AI:** Currently, enemies simply move toward the pill. Adding more complex behaviors (such as pathfinding, targeting the player, or evading projectiles) would create more engaging and challenging gameplay.

**Projectile and Damage System:** Implementing different types of projectiles (e.g., homing projectiles, area-of-effect attacks) or power-ups could introduce variety and strategy in gameplay. Adding a damage type system (e.g., fire damage, freezing) would also add complexity to enemy interactions.

**Health and UI Enhancements:** The health reduction system is basic, and it could be enhanced by adding visual effects (e.g., screen shakes or flashing) to indicate damage. Additionally, improving the UI to display the pill’s health more dynamically and adding sound feedback would make the game more immersive.

**Multiplayer Support:** Introducing cooperative gameplay, where multiple players control different pill stations, could make the game more engaging. This would require significant changes to the spawner and enemy AI systems to handle multiple players effectively.

**Optimizing Performance:** As the number of enemies grows, performance could degrade. Optimizing the projectile and enemy interaction logic, as well as reducing unnecessary object checks (like finding enemies every frame), would improve performance, especially on lower-end devices.