Stellar Sweep

Build a **top-down space shooter** using Unity primitives. A player ship moves and rotates around an arena, shoots bullets at incoming asteroids, and survives as long as possible. You are given **requirements only** — use the in-class demo to figure out implementation.

Submission: Submit link to your GitHub repository

Required Self-Learning — Complete BEFORE implementing

This worksheet requires **Object Pooling** and **Coroutines**, which were not covered in class. Watch these tutorials before you start:

• **Object Pooling:** https://learn.unity.com/tutorial/introduction-to-object-pooling
• **Coroutines:** https://learn.unity.com/tutorial/coroutines

Requirements Checklist

Project Setup

- ■ **3D project**, Unity 6.x or 2021+. If Unity 6+, Active Input Handling set to **Both**.
- ■ Assets folder has organized subfolders: **Scripts**, **Prefabs**, **Materials**.

Scene & Camera

- ■ Arena floor (Plane) with a **material** applied.
- ■ Main Camera is **top-down**, **Orthographic** projection.
- ■ Camera **follows the player** via a script using an offset, updated in **LateUpdate()**.

Player Ship

- ■ Player is a **primitive** (Cube/Capsule) with a material, named and tagged **"Player"**.
- ■ Moves **forward/backward** with vertical input and **rotates** with horizontal input (not sliding).
- ■ Movement uses **Time.deltaTime** and **Input.GetAxis()**.
- ■ Player is **clamped inside the arena** (cannot leave bounds).
- ■ Has a **Rigidbody** (gravity off) and a **Collider**.

Bullets (Object Pooled)

- ■ Bullet is a **prefab** (small Sphere with material). Tagged **"Bullet"**.
- ■ **Spacebar** fires a bullet from the player's position in the player's forward direction.
- ■ Uses **Input.GetKeyDown** (one shot per press).
- ■ Bullets are managed via an **Object Pool** — **no Instantiate/Destroy during gameplay**.
- ■ Bullets **deactivate and return to pool** when leaving bounds or hitting something.
- ■ Has a **Collider** (Is Trigger) and **Rigidbody** (no gravity).

Asteroids

- **3 distinct asteroid prefabs** (different sizes and/or colors) in Prefabs folder.
- Each has: movement script, **Collider** (Is Trigger), **Rigidbody** (no gravity). Tagged **"Asteroid"**.
- Asteroids leaving the arena are **destroyed**.

Spawn Manager (Coroutine-Driven)

- Empty GameObject with **SpawnManager** script handles all asteroid spawning.
- Asteroid prefabs stored in a **GameObject[] array**, assigned via Inspector.
- Random prefab selected each spawn via **Random.Range()**, spawned at **random edge positions**.
- Spawning runs via a **coroutine** (IEnumerator + yield return), **not** InvokeRepeating.
- **Difficulty increases over time** via coroutine logic (e.g., spawn interval decreases, speed increases). Must be **noticeable** after 30 seconds of play.

Collisions

- Bullet hits asteroid: **asteroid destroyed, bullet returned to pool**.
- Asteroid hits player: **Game Over** logged to Console (or player destroyed, spawning stops, etc.).
- Uses **OnTriggerEnter** with **CompareTag**.

Code Quality

- Meaningful variable names, consistent formatting, **comments** on non-obvious logic.
- Variables that don't need Inspector access are **private**.
- No compiler warnings or errors.

Good luck, and have fun with it.