Knockout Arena

Build a top-down arena brawl using Unity primitives. A player sphere rolls around a floating island, enemies chase and try to push you off the edge, and powerups give temporary knockback strength. Waves increase in difficulty. You are given requirements only — use the in-class demo to figure out implementation.

Worksheet: Sound & Effects + Gameplay Mechanics **SWE402: Game Development**

Required Self-Learning — Complete BEFORE implementing

This worksheet requires Delegates & Events, which were not covered in class. Watch these tutorials before you start:

• Events & Delegates: https://learn.unity.com/tutorial/events-uh + https://learn.unity.com/tutorial/delegates

Submission: Submit link to your GitHub repository

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, Physics Materials.

Scene & Camera

- [ ] Arena is an elevated Plane/Cylinder — falling off the edge is the lose condition.
- [ ] An empty "Focal Point" object sits at arena center (0,0,0). Main Camera is a child of it.
- [ ] Focal Point rotates via horizontal input, orbiting the camera smoothly around the arena.

Player

- [ ] Player is a Sphere with a material applied, named "Player", tagged "Player".
- [ ] Has a Rigidbody (gravity ON). Movement via AddForce() with vertical input.
- [ ] Force direction uses focalPoint.transform.forward (not Vector3.forward). Player moves where camera faces.
- [ ] Physics.gravity scaled by a public gravityModifier in Start(). Rigidbody cached via GetComponent<Rigidbody>().
- [ ] Focal point found via GameObject.Find() in Start() — not in Update().
- [ ] A Physics Material (Bounciness >= 1, Bounce Combine = Multiply) applied to Player's collider.

Enemy AI

- [ ] Enemy is a Sphere prefab with material. Tagged "Enemy". Has Rigidbody + same Physics Material as Player.
- [ ] Enemy script: gets player reference in Start(), calculates direction = (player.position - self.position).normalized.
- [ ] Applies AddForce toward the player each Update. Public speed variable controls chase intensity.
- [ ] Enemies that fall below Y < -10 are destroyed via Destroy(gameObject).

Powerups

- [ ] Powerup is a prefab (any primitive) with Collider set to Is Trigger. Tagged "Powerup".
- [ ] Player detects powerup via OnTriggerEnter with CompareTag. Sets hasPowerup = true, destroys powerup.
- [ ] While powered up, OnCollisionEnter with "Enemy" applies ForceMode.Impulse away from player (knockback).
- [ ] Powerup expires after a set duration via Coroutine (IEnumerator + yield return new WaitForSeconds).
- [ ] A visual powerup indicator (child object of Player) is toggled via SetActive(true/false).

Waves & Spawning

- [ ] SpawnManager script on an empty GameObject. Enemy and powerup prefabs assigned via Inspector.
- [ ] A SpawnEnemyWave(int count) method uses a for-loop to instantiate count enemies.
- [ ] Spawn positions generated by a separate method with a Vector3 return type: GenerateSpawnPosition().
- [ ] In Update(), uses FindObjectsByType<Enemy>().Length to detect when all enemies are defeated.
- [ ] When enemy count == 0: waveNumber increments, next wave spawns, and a new powerup spawns.
- [ ] Difficulty increases: each wave spawns more enemies than the previous one.

Animations

- [ ] At least one object has an Animator Controller with 2+ states (e.g., powerup spin, indicator pulse).
- [ ] State transitions are controlled in code via SetBool() or SetTrigger() — not only through the Animator window.

Effects: Particles & Sound

- [ ] At least 1 ParticleSystem that plays on a game event (e.g., powerup collect, enemy knockout). Controlled via .Play() in code.
- [ ] Particle is not set to Play on Awake — it is triggered programmatically at the right moment.
- [ ] Background music via AudioSource on Main Camera. Looped. Volume reduced so SFX are audible.
- [ ] At least 1 sound effect via AudioSource.PlayOneShot(clip, volume) on a game event.
- [ ] AudioClip variables are public (assigned via Inspector). AudioSource cached via GetComponent in Start().

Game Events — Delegates

- [ ] A GameManager (or similar) script declares at least one C# event using a delegate or System.Action.
- [ ] Example: public static event Action OnGameOver; — raised when the player falls off the arena.
- [ ] At least 2 other scripts subscribe to this event and react (e.g., SpawnManager stops spawning, music stops).
- [ ] Subscribers use OnEnable()/OnDisable() to subscribe/unsubscribe from the event.

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.