Code Explanation:

PlayerController.cs

This script controls the main player ball movement and collectible counting mechanism.
In FixedUpdate, it gets horizontal/vertical input and adds force to the Rigidbody, so the ball rolls.
When the ball touches a collectible trigger, it disables that collectible, increases count, updates UI text, and plays collectible chord sound effects.
It also has win-related helper methods: HasEnoughCollectibles() checks if player has collected enough cubes, and WinGame() shows "You Win!" text and starts replaying the music.

The code is written by myself and based on the original roll a ball tutorial.

PlayerAbilities.cs

This script handles player skills.
Right now, it has two skills:

- Flash (Q): teleports forward
- Explosion (E): destroys nearby obstacles only if they have DestructibleObstacle(a class to identify which obstacle can be destroyed).
  It also handles visual effects:
- Flash trail (TrailRenderer burst) which I looked up on the Youtube and watch some tutorials.(References is at the bottom of the doc).
- Explosion visual effect (References is at the bottom of the doc).
  Each ability also triggers the music event.

The code is written by myself with some guidance from CHATGPT to give me some available solutions to make this functionality happen with easy solution. Also Chatgpt and the video tutorials tells me how to make the visual effect easily.

DestructibleObstacle.cs

This script is a marker component to mark obstacles if they can be destroyed or not.
If an obstacle has this script attached, explosion is allowed to destroy it.
If it doesn’t have this script, explosion ignores it.

The code is written by myself.

HoleGoal.cs

This script is attached to the destination hole trigger.
When player step on the hole:

- If player has enough collectibles, it calls WinGame().
- If not enough, it shows "Collect More Cubes First!" for a short period of time.
  It also triggers a “not ready” chord sound effect.

The code is written by myself.

SlopeAccelerator.cs

This script is attached to slopes, in order to have acceleration function when player enter the slope. I made those acceleration settings public therefore we can adjust them later. For example I have 2 type of slopes, one is longer and the other one is wider.

The long slope accelerates slower but have higher max speed.

The wide slope accelerates faster but have slower max speed.

While the player is touching the slope, it calculates uphill direction based on surface normal and adds force to push the ball upward.
It also plays slope chord sound effect with cooldown.

The code is written myself with some guidance from CHATGPT to give me some available solutions to make this functionality happen.

BouncyFeature.cs

This script makes bounce obstacles.

I have made 2 type of bouncy obstacles, one is a pillar and the other one is a small wall.
When player collides with them, it calculates direction away from obstacle and applies forces to push away player.
It resets player velocity before bounce, so the bounce effect feels clean and strong.
It also triggers bouncy obstacle music sound effects.

The code is written myself with some guidance from CHATGPT to minimize the structure.

ProceduralChordSfx.cs

This is the central audio system for chord sounds.

I was having this idea to make the game more musical therefore I want to use piano chord as sound effects. I was searching for sound effect online everywhere but didn't find any resources have a full collection of different chords, therefore I ask CHATGPT to generate for me, and then it gives me this idea of using a existing library.
It generates short chord clips in code (no external audio files needed), caches them, and plays different chords for different actions (collect, flash, explosion, slope hit, etc.).
It also records action timing and can replay the whole recorded sequence at the end.
We also designed a replay progress bar UI for more clear explanation to the player.

The code is written myself with some guidance from CHATGPT to tell me what Unity functions can be used at where.

ObstacleSummonManager.cs

This script spawns random obstacles around Player 1.

This is another mechanism designed for the game. We can have a player 2 just trying to annoy the player 1 by summoning random obstacles and stop the player from reaching the goal hole.
It picks random position in min/max range within a certain range from the player 1 current position.

By making the reference variables public, we can select any multiple obstacle prefabs.
It also keeps prefab rotation by default (so slope angle is kept).

The code is written myself with some guidance from CHATGPT to tell me what Unity functions can be used at where.

SummonerController.cs

This script is for player 2 local input.
When summon key(I set it to "Enter", I make it public so we can change the button later) is pressed, it asks ObstacleSummonManager to spawn an obstacle.
It also has SummonFromButton() so you can connect a UI button OnClick to the same summon logic.

The code is written myself with ChatGPT to refine and minimize.

CameraController.cs

This script makes camera follow player.
At start it saves the offset between camera and player.
In LateUpdate, it sets camera position to player position + offset.

The code is learned from original roll a ball game tutorial, didn't change too much.

CollectibleRotator.cs

This script just rotates collectible objects continuously.
It uses Update and rotates with fixed x/y/z speeds multiplied by Time.deltaTime.

The code is learned from original roll a ball game tutorial, didn't change too much.

References:

1. https://www.youtube.com/watch?v=js2ye2f5iT0
2. https://www.youtube.com/watch?v=adgeiUNlajY
3. ChatGPT
4. https://www.youtube.com/playlist?list=PL-ptF2slHtJAYSWWJ8aqbf1a5tu9u7pAb