Test task for Unity developer - ZigZag

Dear candidate, when completing this test task, we kindly ask you to rely only on your knowledge and experience. Do not involve your friends and acquaintances in the work. After successful completion, we have several more interview stages. Including an interview with our technical lead, where gaps in knowledge and programming skills are always visible. We hope for honest and fruitful cooperation in the future!

Good luck with your implementation!

Description: In the game to be implemented there are 3 game entities - a ball that the player controls, a field along which the ball moves, and crystals located on the field.

The field is a set of square tiles located next to each other on which a ball can roll. If the ball goes beyond the tile into an empty space (where there is no tile), then the ball falls down outside the field and the player loses. If you lose, the game starts again by clicking anywhere on the screen.

The field is a path of n tiles thick, the direction of which can only go straight or to the right. The field is generated randomly ad infinitum, so that a ball can pass through it (there should be no impassable areas or dead ends). At the very beginning of the game, the field is always a square area of tiles measuring 3*3; the generation of a track 1 tile wide begins only after this area.

Each tile can have 1 crystal or none. Let’s conditionally divide the tiles into blocks of 5 tiles each. On each of these blocks 1 crystal should appear. On which tile from the block the crystal will appear is determined by one of 2 rules:
● randomly
● in order. That is, on the first block there is the 1st tile with a crystal, on the 2nd - 2 tiles and so on until the 5th block. Then again from 1 to 5. Which of these two rules in the game will work must be configured by some parameter.

If the ball hits a crystal, the crystal disappears from the tile and the player is considered to have picked up that crystal.

The ball constantly moves across the field at the same speed. The ball speed must be configured by some parameter. The ball can move either straight or to the right. The direction of movement of the ball can be changed by clicking on any part of the playing field. If the ball is moving straight, then after clicking, it will begin to move to the right. After the next click, the ball will start moving straight again.

The camera moves behind the ball so that it is always in the vertical center of the screen. When the player starts the game, the ball stands still and begins to move only after clicking on any part of the screen. The radius of the ball should be approximately 1⁄2 the size of the tile along any of the axes (the tile is either a square or a cube).

The last point is to implement the concept of level complexity. Let's take the following rule: 1. In the simple difficulty level, the thickness of the tile track (n) is equal to 3 tiles 2. In the

average difficulty level, the thickness of the tile track (n) is 2 3. In the difficult level, n = 1 The difficulty level of the game must be configured by some parameter.

Additional tasks:
1. Implement a game using Zenject.
2. If you lose, the level must start again without restarting the scene.
3. Implement a scoring system and display the result on the screen, provided that 1 point is awarded for picking up one crystal. To implement this point, you can use any UI tool of your choice (NGUI, Unity UI, IMGUI, or just do it on regular Gameobjects).
4. Add animations for the disappearance of tiles located behind the ball (those that have already been passed). They can fall down beyond the boundaries of the screen, or go into transparency.

Reference:
As an example of the main mechanics, you should look at the game: on Android: https://play.google.com/store/apps/details?id=com.ketchapp.zigzaggame&hl=ru on iOS: https://itunes. apple.com/ru/app/zigzag/id951364656?mt=8 or game video on youtube: https://youtu.be/9Uqux7wuP0M?t=45s It is important to note that this game does not implement some of the points described in this task.

What is taken into account when assessing:
1. Well-constructed architecture with a focus on game expandability. By extensibility we mean the possibility of potential maintenance of your solution, the ability to make changes to mechanics with minimalchanges to existing code. 2. Clean, readable code.
3. Demonstration of knowledge of OOP patterns and principles.
4. The completed task starts without errors.

What is not taken into account when assessing:
1. Visual component of the completed task. You can find beautiful models/sprites for the task and make beautiful animations, but you can also get by with primitives that are built into Unity3D itself (Sphere, Cube, Capsule).
2. In addition, the visual style of the reference game does not have to be repeated. You can do the task in both 2D and 3D. It is also not necessary to use physics in the solution - moving the ball and going beyond the boundaries can be realized through simple mathematical calculations.
3. Availability of sound and music.
4. Layout and graphic design of UI. Inscriptions a la “You lost” or “Tap to start” are enough.
5. Availability of assemblies for Android or Windows/Mac. It is enough that the game runs in the Unity3D editor.

Test assignment submission format: As a completed test assignment, provide a link to the git repository where you uploaded your solution. You can use any service of your choice - https://github.com/ https://bitbucket.org/product/or any other.