Summary

- The NPC tries to collect as many balls as it can at once in a limited time. Time is shown as a health bar over the NPC character and string on the screen. The balls have three different difficulty levels, each with a reward point based on its difficulty. The game finishes if NPC collects all the balls or the health becomes zero. After that, the NPC returns to base and the game finishes.

  White Ball: Easy, 1 Point
  Orange Ball: Medium, 3 Points
  Red Ball: Hard, 5 Points

  Points can be adjusted in the ball prefabs.

- The NPC uses Unity's NavMesh for pathfinding.
- Balls physics is closed at first since the area is rough and they don't stop. You can enable it from script on the ball prefabs.

GAME

- You can quit the game any time by pressing ESC button.

Start Menu

- There are two input fields. One of them is for the ball count that will be created. If it's empty, it will send the default ball count. The other one is for gameplay time which is also described as health above. It will also send the default gameplay time if it's empty.

- You can see any functioning keyboard buttons if there are any in the gameplay section.

- You can start the game by the start button.

Game Menu

- When you press the start button in the start menu, it will guide you here. You can start the NPC by pressing the "S" button in this menu.

SCRIPTS

- Most of the scripts are understandable. You can find the explanation of the complicated scripts here.

- The NPC has many scripts on it for different purposes. However, NPCController can access all of them and it operates the NPC.

NPCController.cs

- It is responsible for starting the NPC, controlling its behaviours and checking if the game is finished.

- There are many different behaviour states that NPC can operate. They are defined in this class.
- There is a method for adjusting the correct state. When a state is finished, it automatically adjusts to the next one. ChangeState() method handles it.

- SetAction() method is calling the right method for the NPC's state.

NPCDecision.cs

- It is responsible for deciding which ball to collect.

- It has a list of created balls.

- A BallPriority record has been defined in this class. This record contains two pieces of data: Ball and its priority.

- It has a list of BallPriority record.

- CalculatePriority() method iterates all the balls in the ball list. In every iteration, it creates a NavMeshPath and a BallPriority. After that, it checks if there is a path to the current ball in the iteration. If there is, it calculates that path length. After that, that ball priority is calculated by the current health, balls point and distance to that ball with the formula below:

  Priority = (Ball Point / Distance) * Health

  This priority is being added to the priority list with the associated ball. In the end, the priority list is being ordered with the high priority first.

- This class returns the destination with GetDestinationDecision() method.