Takneek Quest (GameDev Challenge)

Documentation (Team Peshwas)

‘Blitz Ball’

Genre- Sports Game

Engine- Unity (2D)

Programming Language- C#

1. Main Menu and Change Scene script

The game when launched starts with the Main Menu scene which has buttons for Play and Quit. Each scene in the game has been assigned an index, for eg the main menu has an index 0. The script ChangeScene.cs is responsible for switching from one scene to another. It uses a variety of functions which take in an int parameter stating the ID of the new scene, where the UnityEngine.SceneManagement namespace has been used. This Script also has the Quit Functionality. On clicking the Quit button, the game window closes.

Each function in the ChangeScene has been uniquely modelled for each button in every scene setting up parameters that later determine the game configuration. This Logic is made using a singleton which is also what the sound object uses to traverse through scenes.

2. Browsing through the various game menus

After a user clicks on the Play button, they are redirected to a new Scene where the game asks them about the game mode. Blitz ball is an AI integrated 2D Unity game which can be played by 1 player as well as 2 players. In all modes, there are 2 teams with 2 players per team. In the 1 player mode, 3 of the other players are AIs, while in 2 player mode, 2 of the other players are AIs. Booleans are used to control the entire logic flow of the code. The Booleans are given values based on what buttons are pressed and on the basis of these Booleans, the Scenes are changed.

If the user chooses the 1 player mode, they are given 4 options of increasing difficulties- Easy, Medium, Difficult and the Extreme mode. Each mode is

differentiated by the speed of the AIs. These modes have been integrated with the code. After this the user is asked about their preferences for the colour of their player. After choosing the colour, the actual game starts.

If the user selects the 2 player mode, the menu is similar with the only addition of another GameType menu. There are 2 types in this mode- Friend and Foe. In the Friend type, both the human players (i.e. the users) will be on the same team facing 2 AIs as their opponents whose difficulty can be chosen. In the Foe type, both users play against each other having an AI each in their teams. Difficulty level is not asked in the Foe type since it is irrelevant to add it since there is an AI on each team. The AIs in this case would both play on Medium difficulty.

Each menu also contains a back button that reverts the user to the previous screen. All these are executed by the MoveToScene() function in the ChangeScene script.

3. The Actual Gameplay

Once the Booleans based on the user preferences are received, the Blitz Ball game starts. The main canvas consists of a clean black background with 4 borders and a central line. Some hidden colliders are also added so that the ball does not stay in a straight line. There is a goal on each side. There is one ball and 4 players. The top menu shows the score and the time.

The game automatically stops when either of the below two conditions are met:

i. Either of the players reaches 5 goals

ii. Gameplay exceeds 5 minutes

Following are the scripts that are executed and their major functionalities:

a. BallMovement.cs: Responsible for the basic parameters of the game and the ball movement specifically. By default, player 1 starts the game.

The Start() function starts the Launch() Coroutine. This Coroutine defines the initial movement of the ball. If player 1 starts the game, the ball moves towards the left and similarly if player 2 starts the game, the ball moves towards the right.

Functions:

1. RestartGame(): A function which sets all the game objects (ball, 4 players) to their default initial positions.

2. MoveBall(): A void type function. Takes a 2D vector as an argument. This 2D vector is the vector along the movement of the ball. The ball starts with an initial speed i.e. ballSpeed (float). After each 'hit', the speed of the ball is incremented, up to a point when it attains a set maximum speed after which the ball moves with this constant speed.

The velocity is obtained by multiplying the ballSpeed with the direction.

3.IncreaseHitCounter(): Increases the hit counter as long as the maximum speed is not attained. After this speed is attained, the hit counter stops running.

b. BallBounce.cs: This script governs the bounce of the central ball. Vectors have been used to store the position vector and velocity of the ball. The HitCounter counts the number of collisions of the ball with the player. This script also defines the final velocity vector of the ball after the collision. It also defines collisions with the borders and the goals.

Functions:

1. Bounce(): It takes a Collision2D argument which detects the name of the Game Object with which the collision has taken place. Based on this name, the x and y coordinates of the Game Objects are updated. The HitCounter is also updated after each collision.

2. OnCollisionEnter2D(): A void type function. Also takes in a Collision2D argument. This function deals with the scoring of goals. Based on the collision with the game object, a goal is scored, accordingly the next player starts the game. In this function, the Launch() coroutine of the BallMovement script is called.

Then the new position and rotation is instantiated on the game object.

c. Player1.cs, Player2.cs, Player3.cs, Player4.cs: These include the Update() function as the main function. This function takes in the input from the keys of the

keyboard. In the Build settings, different Axes are created- two vertical and two horizontal. The velocity of these players are defined using the obtained directions. This also defines the boundaries of the player movement. No player can cross the middle line so as soon as the transform.position.x reaches 0.8 (adjusted due to the radius of the player), it redirects the player to this position with an x coordinate of 0.80.

d. Timemanager.cs: This is a countdown for 5 minutes. When the timer drops down to zero, the game stops and the team with the higher number of points wins the game (Game Over functionality explained later). The Timer Text displays the time left on the top of the screen.

e. PauseMenu.cs: This creates a pause menu on top of the main menu. When the user presses the escape key, the game pauses and the timer stops. This menu is initially disabled in Unity and its functionality is stored in the boolean IsGamePaused.

Functions:

1. Update(): In the update function, the game checks if the escape key is pressed or not. As soon as the escape key is pressed, the pause menu is enabled. Here if the user presses the ‘m’ key, the main menu is displayed and if the user presses the ‘r’ key, the game is replayed. After the ‘r’ key is pressed, the MoveToScene() function is called from the ChangeScript.cs and changes to this same scene and deactivates the pause menu .

2. Resume(): The gameobject this script is attached to (the pause menu), is deactivated. The timescale is set as 1 resuming the timer.

3. Pause(): Pauses the game and sets the time scale to 0 thereby pausing the timer.

4. Game Over: As soon as 5 goals are scored or 5 minutes have passed, the team with the higher number of points is declared the winner. There are 3 scenes- one declaring Team 1 as the winner, one declaring Team 2 as the winner and the third declaring a tie. On the basis of the game result, the respective scene is changed again by the MoveToScene() function. Here 2 buttons are created, one to switch to the Main Menu and another one to Replay.

5. Background Music- Added music traverses through each scene using the concept of a singleton.

6. Automated AI scripts- Uses an attacker AI and Defensive AI if no human player on the team. Else the AI is only defender AI protecting the goal. The difficulty is adjusted by changing the speed of the AI and player. Extreme Also changes Ball Speed for increasing Difficulty and high paced fun game. The defensive AI is modeled on pure RL as it keeps ball out of the goal.

7. Post processing: The Bloom feature is added to each scene. This creates the glowing effect on the various sprites and texts. A post processing layer component has been added to the main camera after adding a post processing volume asset. Then the post processing volume is used to set the intensity of the Bloom effect in Blitz Ball.

REFERENCES

Sprites, Music: Borders and pucks picked from youtube tutorial along with background sound- https://www.youtube.com/watch?v=JZvNFrS7wTM

Sound effects:- FreeSound.org

Fonts:-fontget.com