Speedy The AR Car

A remote controlled AR car game which you can play at comfort of your home through your ios/android phone. Create the self appointed checkpoints at different locations in your room and race your car to these locations to win the game.

- Interactive UI
- Game Missions/Checkpoint Clearance
- Object/Hurdle placement
- Background music for immersive experience
- Distance calculation from certain point (If there is time)

Technology used are: Unity3ds, Visual Studio 2022, Unity Assets Store,

Main Function And their Uses

1. checkPoints

The checkPoints class is responsible for spawning cubes on a detected AR plane. It uses ARFoundation and ARSubsystems to detect the plane and raycast to determine the position for cube spawning. The main functions and variables are as follows:

Functions:

- Start(): Initializes the ARRaycastManager.
- Update(): Checks if a plane is detected and raycasts to find the position to spawn cubes. If a plane is detected, it calls the SpawnCubes() function.

Variables:

- cubePrefab: Reference to the cube prefab that will be spawned.
- car: Reference to the car GameObject.
- cubeCount: Number of cubes to spawn at the start.
- raycastManager: Reference to the ARRaycastManager component.
- hitResults: List of ARRaycastHits to store hit information.
- isPlaneDetected: Flag to track if a plane is detected.

2. control_car

The control_car class controls the movement and rotation of the car. It receives input from the control_joystick class and updates the car's position and rotation accordingly. It also handles collisions with cubes and updates the score. The main functions and variables are as follows:

Functions:

- Start(): Initializes the Rigidbody component and sets drag and maxAngularVelocity.
- Update(): Retrieves joystick input and applies movement and rotation to the car.
- OnCollisionEnter(Collision collision): Handles collision with cubes, destroys them, increments the score, and updates the UI text.
- UpdatePointsText(): Updates the UI text with the current score.

Variables:

- movespeed: The movement speed of the car.
- rotation_speed: The rotation speed of the car.
- drag: The drag value for the car's Rigidbody component.
- rb: Reference to the car's Rigidbody component.
- control: Reference to the control_joystick class.
- points: The current score.
- pointsText: Reference to the UI Text component for displaying the score.

3. control_joystick

The control_joystick class handles the touch input for controlling the car's movement. It detects drag events on a virtual joystick and provides the horizontal and vertical input values for car movement. The main functions and variables are as follows:

Functions:

- Start(): Initializes the joystick and background images.
- OnDrag(PointerEventData eventData): Calculates the joystick input vector based on the drag position and updates the joystick image position accordingly.
- joystick_LR(): Returns the horizontal input value from the joystick or from the keyboard if no joystick input is detected.
- joystick_TB(): Returns the vertical input value from the joystick or from the keyboard if no joystick input is detected.
- OnPointerDown(PointerEventData eventData): Handles pointer down event, triggering the drag event.
- OnPointerUp(PointerEventData eventData): Resets the input vector and joystick image position.

Variables:

- b_img: Reference to the background Image component.
- joystick_img: Reference to the joystick Image component.
- in_vector: The current input vector from the joystick.

4. spawnableManager

The spawnableManager class allows the player to spawn an object on a detected AR plane by tapping on the screen. It uses ARFoundation and ARSubsystems to raycast from the camera and instantiate the spawnable prefab. The main functions and variables are as follows:

Functions:

- Start(): Initializes the ARRaycastManager and finds the AR camera.
- Update(): Checks for touch input and performs a raycast from the camera. If a plane is hit, it instantiates the spawnable prefab at the hit position.

Variables:

- m_RaycastManager: Reference to the ARRaycastManager component.
- spawnablePrefab: Reference to the spawnable prefab to instantiate.
- arCam: Reference to the AR camera.
- spawnedObject: The currently spawned object.
- hasSpawnedObject: Flag to track if an object has already been spawned.

5. Timer

The Timer class manages the game timer and triggers a game over event when the time runs out. It updates the timer UI text and provides a method to restart the game. The main functions and variables are as follows:

Functions:

- Start(): Initializes the current time with the total time.
- Update(): Decreases the current time and checks if it reaches zero. If it does, triggers the game over event.
- UpdateTimerText(): Updates the UI text to display the current time.
- GameOver(): Activates the game over screen and performs additional game over logic.
- RestartGame(): Reloads the current scene to restart the game.

Variables:

- totalTime: The total time in seconds.
- currentTime: The current time remaining.
- isGameOver: Flag to track if the game is over.
- timerText: Reference to the UI Text component for displaying the timer.
- gameOverScreen: Reference to the game over screen GameObject.

6. scoreCount

The scoreCount class keeps track of the player's score and provides a method to add points. It updates the score UI text with the current score value. The main functions and variables are as follows:

Functions:

- AddScore(int points): Adds the given points to the score and updates the UI text.
- UpdateScoreText(): Updates the UI text to display the current score.

Variables:

- score: The current score.
- scoreText: Reference to the UI Text component for displaying the score.

7. cubeCollision

The cubeCollision class handles collision events between the car and cubes. When a collision occurs, it checks if the collided object is a cube and adds points to the score count. The cube is then destroyed. The main functions and variables are as follows:

Functions:

- OnTriggerEnter(Collider other): Handles collision events. If the collided object is a cube, it adds points to the score and destroys the cube.

Variables:

- points: The points to be added when a cube is collided with.

8. GameOverScreen

The GameOverScreen class handles the game over screen UI. It provides a method to load the game scene again when the "Load Game" button is pressed. The main functions and variables are as follows:

Functions:

- LoadGame(): Loads the game scene when the "Load Game" button is pressed.

Variables:

None.

Apologies for the confusion. Here's the documentation for the "ObstacleManager" class:

9 ObstacleManager Class

The ObstacleManager class is responsible for managing the obstacles in the game. It spawns and positions the traps in the game scene and handles game over events.

Properties

- trapPrefab: A reference to the prefab of the trap object to be spawned.
- car: A reference to the car game object in the scene.
- gameOverScreen: A reference to the GameOverScreen script that handles the game over screen.
- trapCount: The number of traps to spawn at the start of the game.

Methods

- Start(): This method is called when the script instance is being loaded. It calls the SpawnTraps method to spawn the initial traps.
- SpawnTraps(int count): This method spawns the traps in random positions around the car. It takes the number of traps to spawn as a parameter. It generates random positions and rotations for each trap using the carPosition and Random class. It instantiates the trap prefab, sets its position and rotation, and adds the TrapCollision component to handle trap collision events.
- GameOver(): This method is called when the game over condition is met. It shows the game over screen or performs any other necessary game over actions by calling the ShowGameOverScreen method of the GameOverScreen script.

Logic And Explanation behind it

1. checkPoints class:
   - Checks if a plane is detected using AR raycasting.
   - Spawns cubes on the detected plane at random positions and rotations.
   - Attaches a cubeCollision component to each cube for collision detection.
2. control_car class:
   - Receives input from the control_joystick class to control the car's movement.
   - Moves the car forward or backward based on the joystick input.
   - Rotates the car based on the joystick input for turning.
   - Handles collision events with cubes.
   - Updates the score count and UI text when a cube is collected.
3. control_joystick class:
   - Captures touch input and calculates the direction based on the position of the touch on the virtual joystick.
   - Provides values for the car's horizontal and vertical input.
   - If no touch input is detected, falls back to keyboard input for car control.
4. spawnableManager class:
   - Waits for a touch input.
   - Performs an AR raycast from the camera to detect a plane.
   - If a plane is hit, instantiates a spawnable prefab at the hit position.
5. Timer class:
   - Tracks the remaining time in the game.
   - Decreases the current time in each frame.
   - Triggers the game over event when the time reaches zero.
   - Updates the timer UI text.
6. scoreCount class:
   - Keeps track of the player's score.
   - Provides a method to add points to the score.
   - Updates the score UI text.

7. cubeCollision class:
    - Handles collision events between the car and cubes.
    - Checks if the collided object is a cube.
    - Adds points to the score count when a cube is collected.
    - Destroys the collided cube.
8. GameOverScreen class:
    - Handles the game over screen UI.
    - Provides a method to reload the game scene when the "Load Game" button is pressed.

These classes collectively implement the game logic for an AR car game, including cube spawning, car control, scoring, time management, collision detection, and game over handling.

Unity Engine:

This is the part where I explain how closely related all the class script are inside unity engine. Also how did they get attached with the objects to make them work as intended.

1. checkPoints class:
    - Attached to a game object in the Unity scene.
    - The cubePrefab and car game objects are assigned in the Unity editor.
    - Uses the ARRaycastManager to perform raycasting and detect planes.
    - Spawns cubes on the detected plane using the cubePrefab at random positions and rotations.
    - Adds the cubeCollision component to each spawned cube.
2. control_car class:
    - Attached to the car game object in the Unity scene.
    - The movespeed, rotation_speed, and drag values are set in the Unity editor.
    - Uses the control_joystick class to receive input for car movement.
    - Applies movement and rotation to the car's Rigidbody component based on the input.
    - Handles collision events with cubes to update the score.
3. control_joystick class:
    - Attached to a UI joystick element in the Unity scene.
    - Uses touch input to determine the direction of the virtual joystick.
    - Calculates input values based on the touch position and provides them to the control_car class.
    - Falls back to keyboard input if no touch input is detected.
4. spawnableManager class:
    - Attached to a game object in the Unity scene.
    - The m_RaycastManager and spawnablePrefab are assigned in the Unity editor.
    - Uses touch input to detect a plane using AR raycasting with the ARRaycastManager.
    - Instantiates the spawnablePrefab at the detected hit position on the plane.
5. Timer class:
    - Attached to a game object in the Unity scene.

- The totalTime value is set in the Unity editor.
- Updates the timer text UI element with the remaining time.
- Triggers the game over event when the timer reaches zero.

6. scoreCount class:
   - Attached to a game object in the Unity scene.
   - Updates the score text UI element with the current score.
   - Provides a method to add points to the score.
7. cubeCollision class:
   - Attached to each spawned cube game object.
   - Handles collision events with the car.
   - Checks if the collided object is a cube and adds points to the score.
8. GameOverScreen class:
   - Attached to a game object representing the game over screen UI.
   - Provides a method to load the game scene when the "Load Game" button is pressed.

These classes are utilized within the Unity engine by attaching them to game objects, assigning required references, and using their methods and properties to implement the game logic, handle user input, update UI elements, and manage game states.

Pictures of Unity Assets: