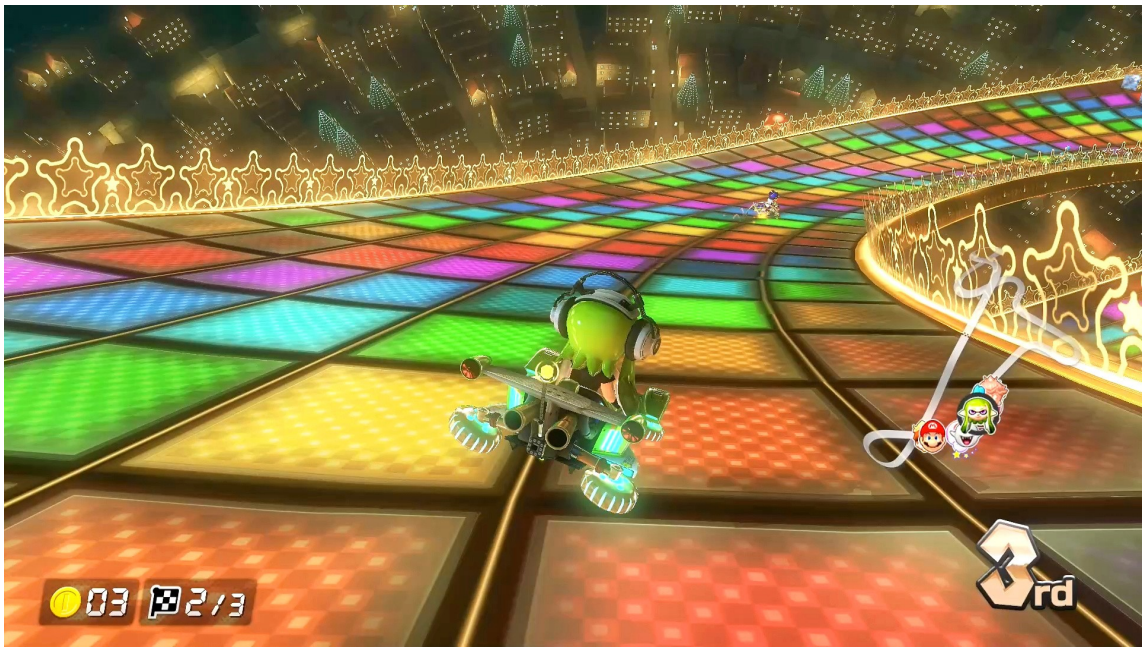
Kevin Kwan, Shantanu Bhatawadekar
2/25/2020
Game Design 4th Period

Evade and Deceive: Final Design Document

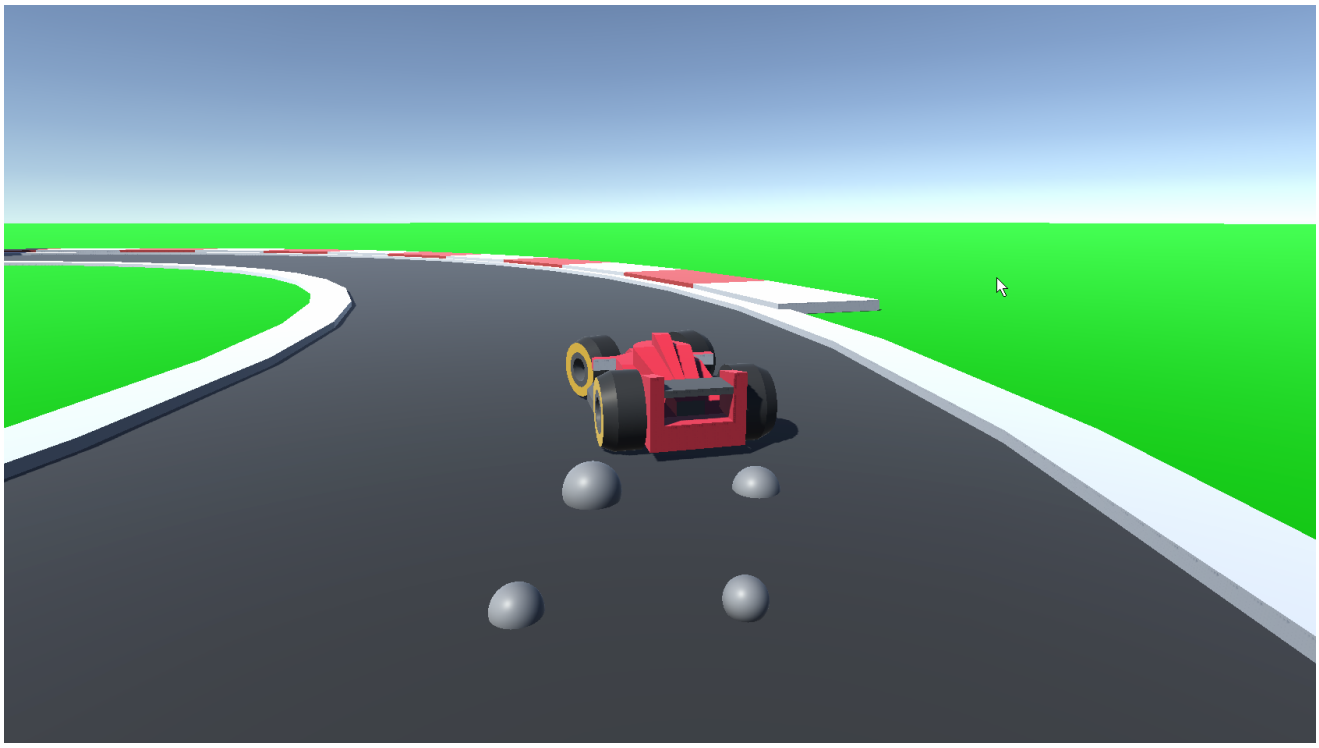
Disclaimer: Our game is still in development and considered incomplete according to what we aspire for the final game to be as we discussed in our game proposal. For now, we have a completely functional, single and multiplayer racing game with powerups that still follows the overview of vision of our game, but it is currently lacking the complicated machine learning aspect that is the police. Please read our design document with this in mind as we discuss our game design thus far and all the things that we have completed.

Overview

Evade and Deceive is a competitive, singleplayer and multiplayer, third-person 3D racing game with a similar playstyle to Mario Kart. The player will traverse and race through different levels and maps varying in different themes with the use of power-ups and driving skills to be the first to arrive at a specific final destination in the shortest amount of time. However, certain hindrances like the walls, buildings, environment debuffs, minor track obstacles, and even other players will impede your path, preventing you from reaching your goal. Get creative, discover the fastest path, control your movement, dodge enemy projectiles, use your power-ups to your advantage, and challenge yourself and other players online in order to reach your destination in the fastest time possible!



This game follows a similar racing playstyle to Mario Kart. However, the player will be chased by police cars and the graphic style will be more low poly. There will be elements, hazards, and other obstacles implemented in the level design of the levels that the player can use and avoid in order to successfully evade the police.



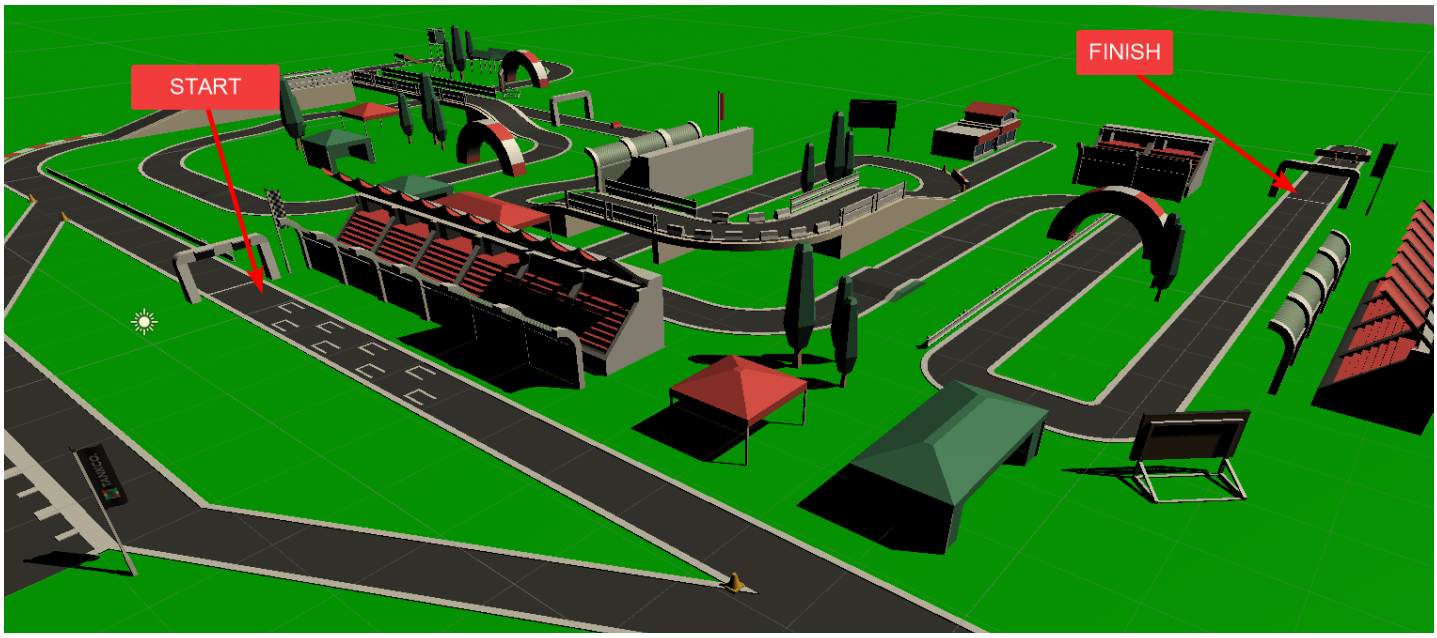
This screenshot displays the very early prototype and idea for basic racing functionality in our game. Particle effects, car model, and level design were all subject to change. This basic movement and prototype model allowed us to continue developing ideas and visualizing how the final product will look.

Genre

The genre of our intended game is a competitive, singleplayer and multiplayer, third-person 3D racing game that borrows the driving and playstyle from Nintendo's Mario Kart series. The player must use their decision-making and racing skills in order to drive through a racetrack disguised as an obstacle course as quickly as possible to reach the finish line. The player wins by simply reaching the end goal, but competition is apparent when the player races against themself or against other people to finish the race first and get the best time. If the player gets stopped by obstacles or does not drive fast enough, their race continues ticking or they can fall behind other players in terms of race position. The player's control of their driving movement and usage of their power-ups is crucial in order to gain advantages over players and skillfully traverse the map efficiently. There are many paths that the player can use to get to their specific destination so the player can get creative and choose the best and fastest way for them to get there. Of course, there will be obstacles that will try to hinder the player from reaching their destination.

Theme

The theme of the game is variable per sets of levels, meaning that a couple of levels will share the same themes and settings before transitioning to new themes and settings as the levels progress forward. Settings can include a farm, a highway, a suburb, a town, a racetrack, a rainbow road in space, and more. The maps and the levels will be designed to be semi-realistic based off of locations in the real-world, but designed in a way such that each level is a very well-disguised racetrack. For example, think about the player driving into a farm, driving on a dirt road while dodging farm-related obstacles with the police behind them, and then they manage to reach a country road at the end of the track and drive into the sunset to evade the police in this level. The main setting of the game will be conceptually a racetrack that the player is on and will drive around in order to reach the finish line. However, the racetrack is also designed in such a way to be compared similarly to an obstacle course. Regarding what we have done, the first couple of levels are centered around the theme of a generic racetrack, but built differently in each level, such that the player can first grasp the idea that the game is focused on racing. Later on, the levels will have other themes such as space, so the player will continue to hold the idea that they are racing but they are now in outer space or in some other setting.



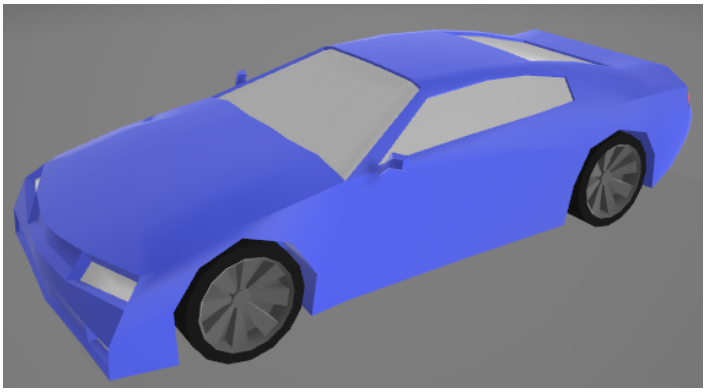
This is the original track for level 2 of the game, which is still under the theme of a generic racetrack. The start and finish of the track have been labeled, and this is the track that players must traverse through.

Visual Aesthetics

The game has a semi-realistic aesthetic of the modern world but in a low-poly style with textures, lighting, and reflections based off of the looks of the real-world, as seen in the screenshot earlier of map 2. Depending on the theme of the level, extra lighting effects or colors may be included. Future maps with different settings such as outer space will be implemented later. The game is also played in a very smooth and satisfying third person view to allow for observation in the 3D world with the use of smart Cinemachine Cameras. The maps are designed in such a way to be simple, visually appealing, identifiable, and not super distracting while distinguishing the track for the player to drive on. While everything will be low-poly, they will be detailed enough for the player to identify what the object or surroundings are supposed to represent.

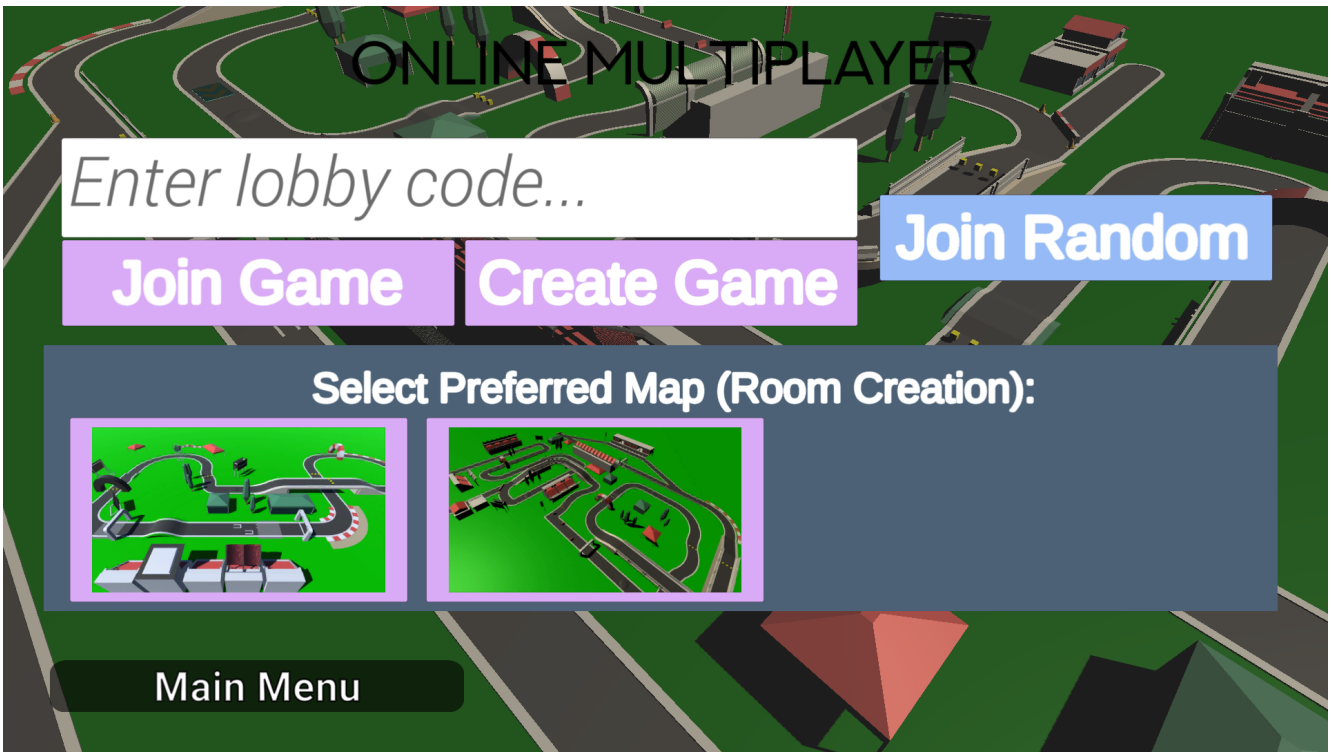
```
GameObject[] cams = GameObject.FindGameObjectsWithTag("camcam");
GameObject cameraview = this.transform.Find("CM vcam1").gameObject;
for (int i = 0; i < cams.Length; i++)
{
    if (cams[i].layer == 12)
    {
        p1 = true;
    }
    if (cams[i].layer == 13)
    {
        p2 = true;
    }
    if (cams[i].layer == 14)
    {
        p3 = true;
    }
    if (cams[i].layer == 15)
    {
        p4 = true;
    }
}
//basically checks all cameras to see if they have layers on them

if (!p1) //if not, start assigning
{
    cameraview.layer = LayerMask.NameToLayer("P1");
    PlayerCamera.GetComponent<Camera>().cullingMask &= ~(1 << 13);
    PlayerCamera.GetComponent<Camera>().cullingMask &= ~(1 << 14);
    PlayerCamera.GetComponent<Camera>().cullingMask &= ~(1 << 15);
}
else if (!p2)
{
    cameraview.layer = LayerMask.NameToLayer("P2");
    PlayerCamera.GetComponent<Camera>().cullingMask &= ~(1 << 12);
    PlayerCamera.GetComponent<Camera>().cullingMask &= ~(1 << 14);
    PlayerCamera.GetComponent<Camera>().cullingMask &= ~(1 << 15);
}
else if (!p3)
{
    cameraview.layer = LayerMask.NameToLayer("P3");
    PlayerCamera.GetComponent<Camera>().cullingMask &= ~(1 << 12);
    PlayerCamera.GetComponent<Camera>().cullingMask &= ~(1 << 13);
    PlayerCamera.GetComponent<Camera>().cullingMask &= ~(1 << 15);
}
else if (!p4)
{
    cameraview.layer = LayerMask.NameToLayer("P4");
    PlayerCamera.GetComponent<Camera>().cullingMask &= ~(1 << 12);
    PlayerCamera.GetComponent<Camera>().cullingMask &= ~(1 << 13);
    PlayerCamera.GetComponent<Camera>().cullingMask &= ~(1 << 14);
}
```

This code snippet handles the assignment of cameras to each individual camera in online multi-play through the use of culling masks and layers and by checking the existence of assigned layers and players and assigning their corresponding designation of layer being P1, P2, P3, or P4. The assignment continues to work even if a player joins and leaves and another player joins in their place. The result is that each player will see through their own camera and their own controllable car, despite there being multiple cameras in the game scene.



Low-poly model of the racing car in our game.

The UI is also very clean and easy to navigate. Below are screenshots of the main menu and the multiplayer menu screen.

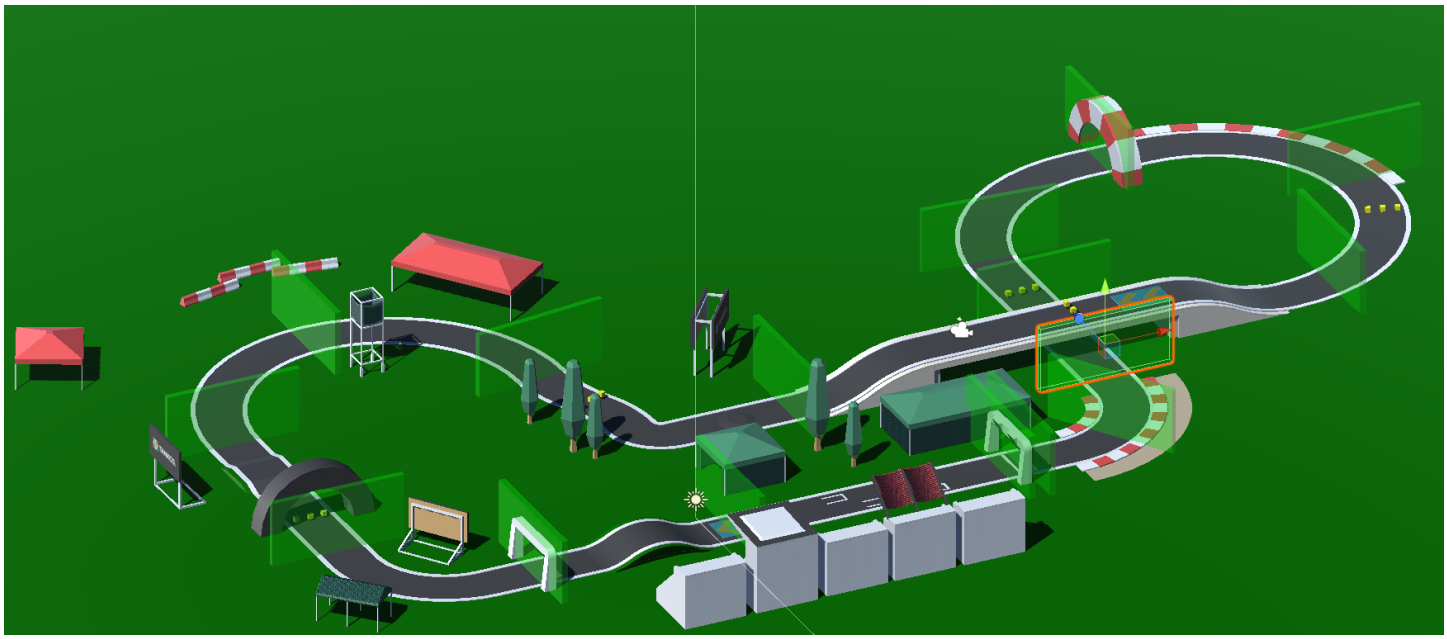


The UI in-game is also very clean, displaying lots of helpful information with a minimap, a timer for how long it takes for the player to complete the race, what their best time on the current map is, what the room code is, what position in the race the player is in compared to everyone else, and how many shells they have.

The following is a pause menu that shows up in game when the player presses Escape on their keyboard. It allows them to resume playing the game, return to the main menu and leave the race, and adjust the current volume of the game using a slider. They can press Escape again to hide this menu.

Goal

In each level, there will be a different goal and map where you have a start point and you must reach the finish line as quickly as possible while dodging obstacles and beating other players to the finish line. You must skillfully drive through the map, avoid as many obstacles as you can, and prevent yourself from being beaten in the race by other players. The racetracks are not like regular racetracks where the player must drive in a loop and reach where they started to finish the race. They must drive from point A to point B, where the latter symbolizes the finish line. There are invisible checkpoints in between these two points that the player must go through in order to check if they cheated to reach the end without driving on the track. If they pass all the checkpoints and finish the race, their finish will be legitimized and the completion of the race will be counted. You can use the knowledge of the map and the help from power-ups to aid you in reaching the goal and interfere with other players. By replaying the level, players can gain more knowledge regarding the layout of the map, take into account the location of obstacles, where the best places to use power-ups are, and more. The player will win the level once they have reached the level's finish line. The player will win the game once they have completed all the levels and beaten the final level, which will be the most difficult of all.



This screenshot displays the checkpoints that are placed along the track (visible here, but will not be visible in the build of the game) to check to make sure that the player is not driving off the track to cheat the race. They must drive through all the checkpoints in the correct order in order for their completion of the track to be counted.

```
public void PlayerThroughCheckpoint(CheckpointSingle checkpointSingle)
    {
        items = new List<ITEM>();
        GameObject[] cars = GameObject.FindGameObjectsWithTag("car");
        if (started==false&&checklist.IndexOf(checkpointSingle) == 0)
        {
        timer.Start();
        started = true;
        }
        if (checklist.IndexOf(checkpointSingle) == nextCheckpointSingleIndex)
        {
            checkAmount++;
            TrackCheckpointsUI yeppers = GetComponentInChildren<TrackCheckpointsUI>();
            brotherman.Hide();
            LastCheckpointSingleIndex = checklist.IndexOf(checkpointSingle);
            nextCheckpointSingleIndex = (nextCheckpointSingleIndex + 1) % checklist.Count;
            if (!finishrace)
            {
                foreach (GameObject car in cars)
                {                    items.Add(new ITEM { itemObj = car, startPos =
car.GetComponent<ThirdPersonCharacterController>().checkAmount });
                }
                items = items.OrderByDescending(w => w.startPos).ToList();
                for (int i = 0; i < items.Count; i++)
                {
               items[i].itemObj.GetComponent<ThirdPersonCharacterController>().posInRace = i + 1;
                }
            }
            if (checklist.IndexOf(checkpointSingle) == checklist.Count - 1&& !finishrace)
            {
                for(int i = 0; i < items.Count; i++)
                {
                    items[i].itemObj.GetComponent<ThirdPersonCharacterController>().posInRace = i +
1;
                }
                finishrace = true;
                brotherman2.Show();
                brotherman2.GetComponentInChildren<TextMeshProUGUI>().text =
"Congratulations!\n~"+formatters(posInRace)+" place~";
                timer.Stop();
                checkAmount+=1;
                showthepositionplace.SetActive(false);
                float timeTaken = (float)timer.Elapsed.TotalMilliseconds;
                if (sceneBestTime == 0 || timeTaken < sceneBestTime)
                {
                    PlayerPrefs.SetFloat(key, timeTaken);
                    besttime = TimeSpan.FromMilliseconds(PlayerPrefs.GetFloat(key));
                }
                Cursor.lockState = CursorLockMode.None;
            }
        }
        else if (checklist.IndexOf(checkpointSingle) == LastCheckpointSingleIndex)
        {
            brotherman.Hide();
        }
        else if (started == false&& checklist.IndexOf(checkpointSingle) == checklist.Count - 1)
        {
            brotherman.Hide();
        }
        else
        {
            if (!finishrace)
            {
                OnCorrectCheckpoint?.Invoke(this, EventArgs.Empty);
                brotherman.Show();
            }
        }
}
```

Here is a snippet of the code in the player's controller script associated with the checkpoint system implemented. This method detects when the player crosses the right and wrong checkpoints whenever they cross one and ensures that they will be able to finish the race if they pass all the checkpoints in the correct order.

A warning will show up above on their screen whenever they miss a checkpoint, telling them to press R to return them to the last checkpoint they have correctly crossed.

Sound Design

The game will feature noncopyright or licensed (courtesy of Monstercat), empowering, upbeat, pumping background music for different themes of levels. Shantanu has experience in producing beats and instrumentals, so he is able to provide non-copyright or his own licensed music for the background music of our game. The game will have sound effects such as engine sounds, driving, collision sounds such as when a player gets hit by a shell, notification sounds such as when a player has crossed a checkpoint or finished the level, and more. In multiplayer, the music also plays individually for each player.

Mechanics

The main mechanic of this game is the controlled movement through driving a simple car such as driving forward, turning left and right, reversing, braking, and combinations of all of these in order to beat racing levels. The driving of the player's car is what allows the player to move in each level. The player has ultimate control of where to move, so they must use their skills and knowledge of the level's layout with the help of power-ups to drive efficiently through a level and reach the end. The power-ups that they can gather, identified as yellow

collectable dice placed in rows at certain intervals on the track, can allow the player to shoot projectiles to hit other enemy cars to slow them down. There are also boostpads and slowpads that speed up or slow the player down when they drive across them. The player is also given a minimap on their HUD to see the locations of nearby cars, a bird's eye view of the surrounding area, and where power-ups are located. The HUD will also display their current time, their best time, whether or not they missed a checkpoint or finished the race, the room code, and what race position they are in.

Controls

The keyboard controls are W, to drive forward, S, to drive backward (reverse, also serves as braking), A, to turn left, D, to turn right, and space to use a powerup. The camera will be following the player, so the mouse will not be used during gameplay. The mouse cursor will be used in the Main Menu, Level Selection, and Multiplayer screen in order for the player to click and select certain options such as starting up or joining a multiplayer lobby. The keyboard will be used to type in input fields such as when setting one's desired username, creating a lobby by coming up with a room code, and inputting a room code given by a friend to join an existing room. In any level, hitting ESC will bring up a pause menu where the player can use their mouse to click to resume the game, go back to the main menu, or adjust the volume of the game. Hitting SPACE will make the player use a powerup that they collected, such as shooting a shell if they collected one from hitting a dice. If the player misses any checkpoints or if they somehow become stuck, pressing R will allow for the player to return back to the most recent correct checkpoint that they have passed.

Players

This game will be mainly available for single-player and online competitive multiple-player. The player can get a feel for the maps, learn how to drive, play the campaign, and challenge themselves by playing single player in order to practice the maps and get good at completing them. However, in multiplayer, you will be able to challenge other players in real life to race against them with the objective of finishing the race before them. Power-ups such as the projectile can be collected and used by players to shoot and slow other players down. Players can use different computers to connect online, and play levels together while competing against each other. Currently, each level will allow for a max of 4 players for online play. Multiplayer functionality has been achieved using Photon Unity Networking for online multiplayer.

This screenshot displays the implementation of Photon Unity Networking to allow for multiplayer gameplay functionality between players on different computers. Here, I am racing against two other people, and we are all playing over the internet. The checkpoints continue to work and the players' positions and their models are synchronized with the use of a server and the Photon library. All players can control their own individual cars.

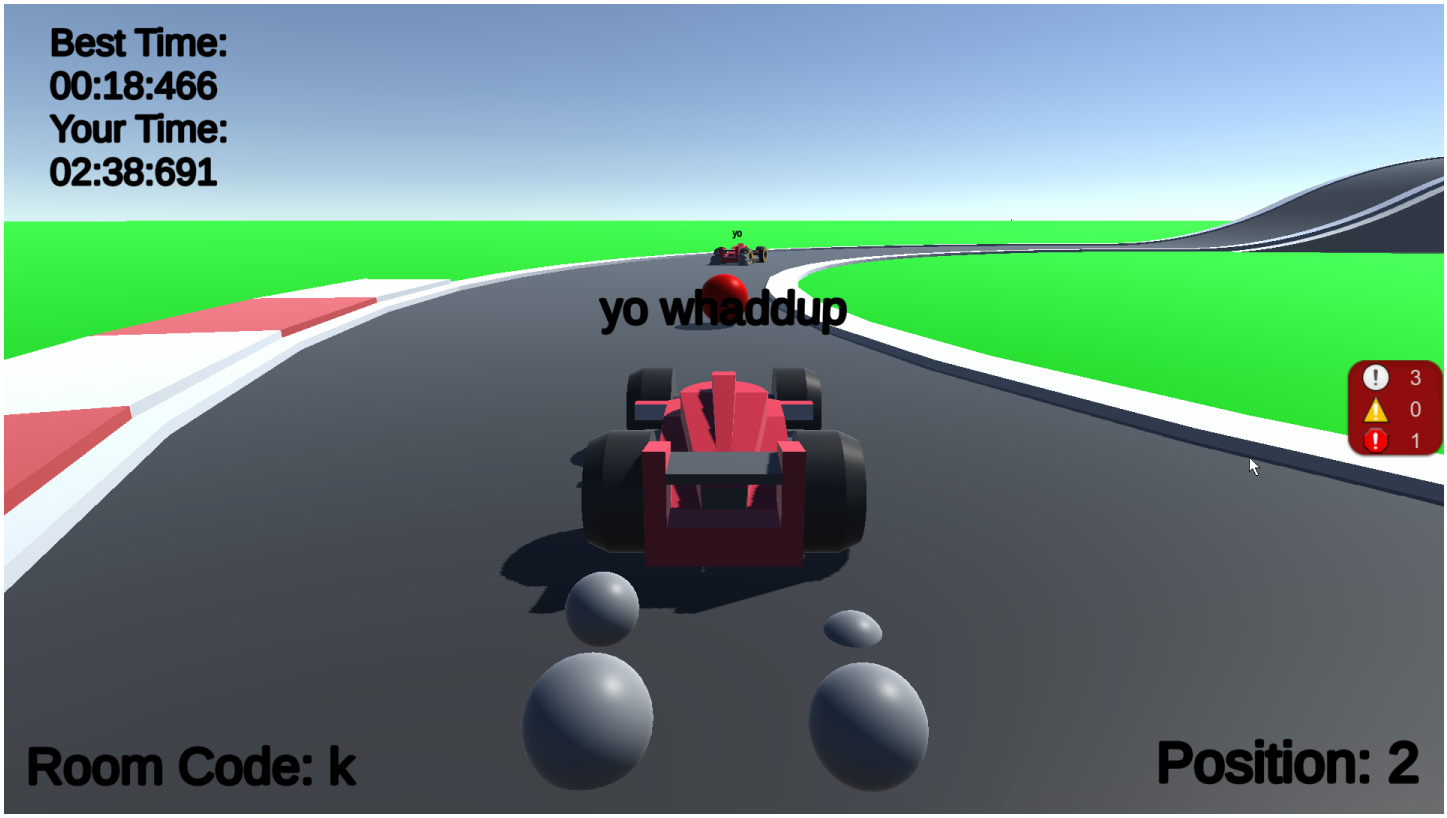
There is also a single-player option in the game where the player can select which map to play on and race by themselves, which also utilizes Photon Engine. Basically, the player selects a map to play, which creates a new private room with a max player count of 1, such that only that player can be in the room racing around until they decide to leave. Currently, there are only 2 maps, but more will be added later.

The following code snippet shows some of the methods that correspond to the buttons on the multiplayer menu to perform some of the actions that I just mentioned with the help of Photon.

```
void Start()
    {
        PhotonNetwork.AutomaticallySyncScene = true;
        Text yuh=prefillname.GetComponent<Text>();
        string name = PhotonNetwork.NickName;
        if (PhotonNetwork.NickName != "")
        {

            Username.text= name;
        }

        usernameMenu.SetActive(true);
        if (!PhotonNetwork.IsConnected)
        {
            Debug.Log("Connected");
            PhotonNetwork.ConnectUsingSettings();

        }

        errorcool.SetActive(false);
        error1cool.SetActive(false);

    }
public void SetUserName()
    {
        usernameMenu.SetActive(false);
        PhotonNetwork.NickName = Username.text;
    }
    public void SetAsPastUserName()
    {if (PhotonNetwork.NickName != "")
        {
            usernameMenu.SetActive(false);
            PhotonNetwork.NickName = PhotonNetwork.NickName;
        }
    else
        {
            error1Message.text = "This is your first time playing! You never came up with a name!
Please set a username first.";
            error1cool.SetActive(true);
        }
    }
    public void CreateGame()
    {
        if (CreateGameInput.text == "")
        {
            errorMessage.text = "Failed to create a room! Did you forget to choose the lobby name?";
            errorcool.SetActive(true);
        }
        else
        {
            PhotonNetwork.CreateRoom(CreateGameInput.text, new RoomOptions() { MaxPlayers = 4 },
null);
        }
    }
public void JoinGame()
    {
        if (CreateGameInput.text == "")
        {
        errorMessage.text = "Failed to create or join a room! Did you forget to input the lobby
    name?";
        errorcool.SetActive(true);
        }
        RoomOptions roomOptions = new RoomOptions();
        roomOptions.MaxPlayers = 4;
        PhotonNetwork.JoinOrCreateRoom(CreateGameInput.text, roomOptions, TypedLobby.Default);


    }
public void JoinARandomGame()
    {
        RoomOptions roomOptions = new RoomOptions();
        roomOptions.MaxPlayers = 4;
        PhotonNetwork.JoinRandomRoom();
    }
public override void OnJoinedRoom()
    {
        PhotonNetwork.AutomaticallySyncScene = true;
        Scene scene = SceneManager.GetActiveScene();
        Debug.Log("Active scene is '" + scene.name + "'.");
        PhotonNetwork.LoadLevel(levelname);
    }
```

Powerups

Powerups will be collectable items in the shape of the mystery boxes from Mario Kart. These will be scattered all around the levels along the racetrack. When the player touches one of the mystery boxes, they will collect the power-up which gives them a random positive effect that helps them in some way. The player can have only one positive effect in their inventory that they can use by pressing space after collecting a power-up. After using a power-up, their inventory will be cleared and let them collect another powerup. The power-ups that they can gather can help speed up their car temporarily, shoot projectiles to hit other enemy cars to slow them down, make the car temporarily invincible to prevent slowdowns and getting stuck, and more.



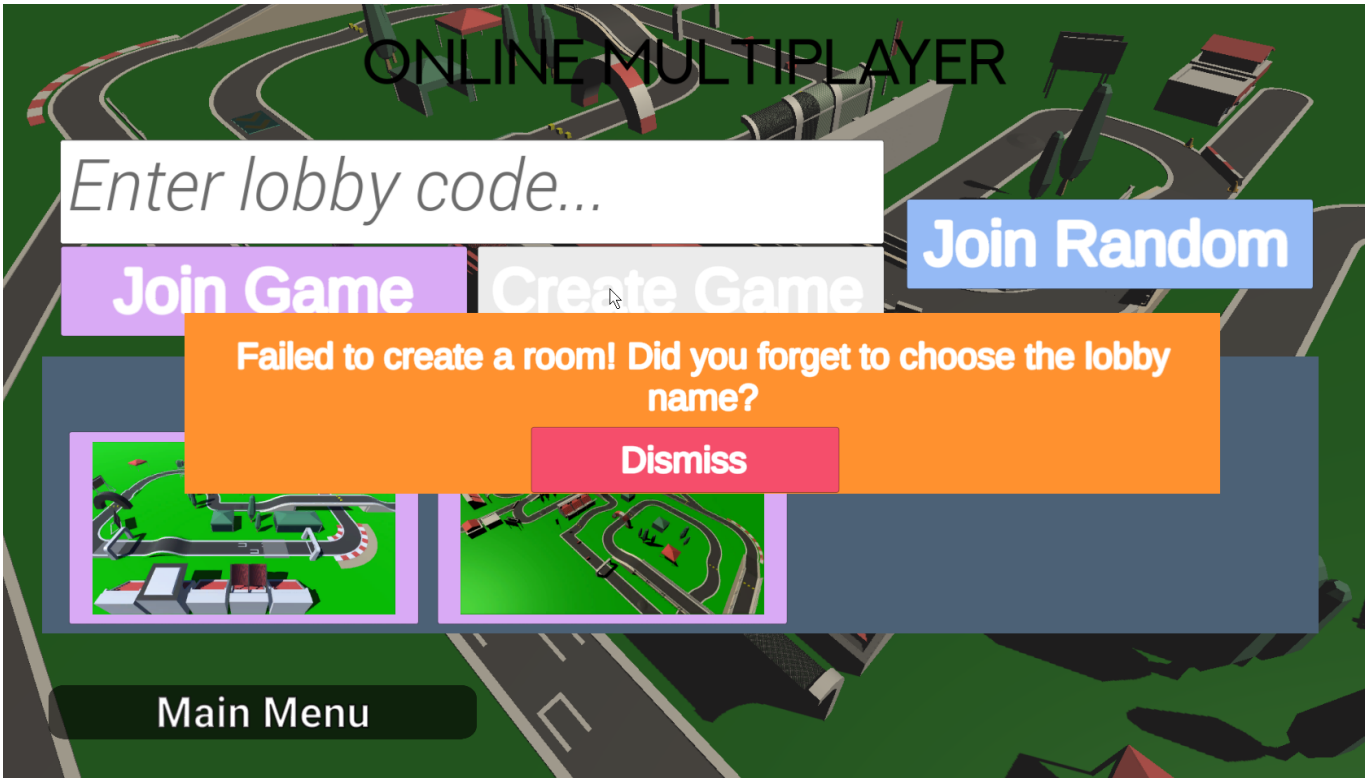
This screenshot shows the current player shooting another player with a red ball projectile. If the projectile hits the car, the other player will be slowed down for a few seconds before regaining normal speed. These powerups can be collected by hitting yellow boxes that are placed around the track that gives the player 1 projectile to use. The player can shoot using SPACE if they have any available balls.

Enemies/Hazards

Enemies and hazards in this game include other cars and players on the track, obstacles relevant to the certain theme of a level, and more. The map is also technically an obstacle itself as it relies on the player's skills to control the driving in such a way to prevent themselves from colliding into walls, figure out where your destination is in the levels, when to use power-ups, perform difficult maneuvers, and determine what the best, fastest path to get to the destination is. Colliding with hazards and obstacles can cause the player to stop like hitting a wall or slow the player down. This inconvenience may create enough time for other players to drive past the player and move ahead. Other players can also act as enemies by shooting projectiles, which are hazards, with the use of a powerup that can hit the player and slow them down for a short duration.

Some Other Miscellaneous Features

Implementation of Error Messages



In this example, the player tried to create a lobby even though they have not entered their desired room code. Thus, this error message pops up that notifies the player telling them to come up with a room code.

Implementation of waiting for enough players to join before starting the race.

Here, we can see that when there is only one player in a multiplayer lobby, the race will not start until there are at least two players ready to race. Once there is enough, up to a max of 4 players, there is a countdown before the race starts (which resets whenever a new player joins to allow for players to get ready) that will start the race once the countdown finishes.

Group Assignment Table (Subject to Change):

Student	Core Responsibilities
Kevin Kwan	ML Agents/AI/Machine Learning, Coding behavior for player movement and enemy behavior, Multiplayer Online Functionality, Coding and design for main menu and level select, UI
Shantanu Bhatawadekar	Music Production, Sound Design, Level Design and Layout, Coding behavior for player HUD, Coding behavior for level objects, Coding behavior for racing power ups, 3D Modeling, advertisement (video)

Technical References

Mario Kart 8 (Screenshot) - https://www.instant-gaming.com/en/2615-buy-game-nintendo-mario-kart-8-deluxe-switch/

Frames Per Second Counter in Unity - http://wiki.unity3d.com/index.php/FramesPerSecond

Creating a Game with Learning AI in Unity! (Tutorial / Machine Learning) - https://www.youtube.com/watch?v=gYwWolRFt98

HOW TO MAKE A DRIVING & RACING GAME IN UNITY C# TUTORIAL BEGINNER/INTERMEDIATE [FULL COURSE] - https://www.youtube.com/watch?v=ehDRTdRGd1w

Kart Racing Game with Machine Learning in Unity! (Tutorial) - https://www.youtube.com/watch?v=i0Vt7l3XrIU

Unity Scene-Specific High Scores using PlayerPreferences - https://answers.unity.com/questions/1447020/scene-specific-high-score.html

Brackeys: START MENU in Unity - https://www.youtube.com/watch?v=zc8ac_qUXQY

Textures/Materials - https://opengameart.org/textures/all

Unity Documentation - https://docs.unity3d.com/Manual/index.html

Unity Play Music Across Scenes - https://answers.unity.com/questions/1392654/background-music-loops-without-restarting-on-playe.html

Background Music - https://lickd.co/

3D Modeling Software - https://www.blender.org/

Arcade Car Driving in Unity - https://www.youtube.com/watch?v=cqATTzJmFDY

Simple Checkpoint System in Unity - https://www.youtube.com/watch?v=IOYNg6v9sfc

HOW TO MAKE AN ONLINE MULTIPLAYER GAME - UNITY EASY TUTORIAL - https://www.youtube.com/watch?v=nmPukdOsYQA

How to install Unity ML Agents Release 8 - https://www.youtube.com/watch?v=aHmkL0Wl9uU.

ML-Agents 1.0+ Creating a Mario Kart like AI - https://www.youtube.com/watch?v=n5rY9ffqryU

ML Agents - https://github.com/Unity-Technologies/ml-agents

FL Studio - https://www.image-line.com/

Photon Unity Networking - https://assetstore.unity.com/packages/tools/network/pun-2-free-119922

Photon Unity Networking - https://www.photonengine.com/

Cinemachine Cameras - https://unity.com/unity/features/editor/art-and-design/cinemachine

Cloud Skybox - https://github.com/keijiro/CloudSkybox

Nitro Fun & Hyper Potions - "Checkpoint" provided by Monstercat by License*

CloudNone - "Speedrun" provided by Monstercat by License*

***Due to the sensitivity and privacy of person information, please contact Kevin Kwan at Koolkev246@gmail.com or KevinKwan2021@gmail.com if you wish to view the signed agreement/proof of license.**