Game Design Document for:

Krash Kart

The Mess of Mazomski Invasions

Written by
Christopher Rojas 301189137
Denisjann Reyes 301249768
Yiu Yiu Yoyo Ho 301256477
Version #01
Dec 01, 2024

CENTENNIAL COLLEGE

TABLE OF CONTENTS

VERSION HISTORY

- Version1.0 - 01 Dec, 2024

GAME OVERVIEW

Philosophy

Philosophical point #1

This game is about pushing the boundaries of traditional car racing games. We're aiming to deliver an experience that's fast-paced, competitive, and filled with chaotic moments where players will have to outsmart their enemies and their competition. Our goal isn't just to race but to deliver a thrilling experience where strategy and reflexes matter. We're taking a classic idea—racing to the finish line—and adding our own twist with pickups, powerful items, and relentless enemies. It’s not about changing the world, but offering a fresh spin on a well-loved genre.

Philosophical point #2

Our game will run on PC. It's about making sure the players are immersed in a world where everything has purpose. We believe that gaming should be accessible and fun without requiring a high-end machine, so we’re optimizing for performance while still ensuring the game looks great. It's not the world that's ending—it's just the need to overcomplicate things.

Philosophical point #3

At the core of this game is a focus on competition. Players must race to the finish line while managing pickups, avoiding attacks, and outsmarting NPC enemies. The design philosophy here is simple: it's all about having fun while competing in intense racing scenarios.

Common Questions

What is the game?

This is a 3D car racing game for two players. Players race to the finish line, competing for speed while picking up power-ups like speed-up, protection, and health reward. The game also features aggressive NPC enemies who relentlessly follow and attack the players, adding an extra layer of challenge.

Why create this game?

We’re creating this game because we believe in the thrill of competitive racing combined with unexpected challenges. While traditional racing games focus purely on speed, we’re adding the unpredictability of power-ups and enemies to keep things exciting. There’s a market for a fast, action-packed racing game with a bit of a twist, and we want to bring that to life.

Where does the game take place?

The game takes place in a post-invasion world, where the skies have turned strange colors, signaling an otherworldly takeover. The environment is filled with chaos, as the hostile enemies—roam and patrol the area, adding danger to every corner. Players must navigate through these treacherous landscapes, racing through environments already scarred by the ongoing invasion. Whether it's navigating city streets or racing across rugged terrains, the world is a battlefield, and every race is a fight for survival.

What do I control?

Players control their own race car with "WASD" and arrow keys respectively, and they can also use Ctrl key to utilize power-ups to enhance their abilities. The game offers commonly used control schemes to accommodate all level players, ensuring an easy-to-learn but hard-to-master experience.

How many characters do I control?

Players control one car each in a two-player mode, but they also interact with the NPC enemies that attack and pursue them during the race. Players must manage both their car's abilities and deal with the hostile NPCs as they race towards the finish line.

What is the main focus?

The main goal is to outpace each other and be the first to reach the finish line. Along the way, players must pick up items that can help them speed up, protect themself from attacks, or refill their health. The NPC enemies add another layer of challenge, as they attempt to sabotage players' progress.

What's different?

This game differentiates itself by blending traditional racing with RPG-like mechanics—pickup items that provide strategic advantages, NPC enemies that add complexity, and a focus on competitive play. Instead of a pure racing simulator, it's a more dynamic, action-filled experience with a focus on outsmarting both the other player and the aggressive enemies.

FEATURE SET

General Features

- Easy controls: The game offers intuitive controls for players of all skill levels, ensuring both newcomers and seasoned gamers can enjoy the experience without a steep learning curve.
- Challenging racing tracks: Tracks are designed to be fast-paced and filled with obstacles, sharp turns, and unique challenges, keeping players on the edge of their seats.
- 3D graphics: The game features immersive 3D environments, bringing the world of the racing game to life with detailed visuals and dynamic animations.

Multiplayer Features

- Up to 2 players: The game supports two-player local multiplayer, allowing friends or family to race against each other in competitive, fun-filled gameplay

Gameplay

- Fast-paced racing: Players race to the finish line as quickly as possible, with each track presenting new obstacles and challenges.
- Power-ups: Collect power-ups like speed ups, protection against attacks, and health rewards to stay ahead of the competition.
- NPC enemies: Enemies with different states patrol the tracks, relentlessly pursuing players and adding an extra layer of difficulty and chaos.
- Track variety: Players race through a variety of different environments, such as villages, forests, and mountains, each with its own set of challenges and strategic elements.
- Competitor interaction: Players can sabotage each other with attacks or clever use of pickups, ensuring that every race is a mix of skill and strategy.
- Health system: Keep an eye on the health during the race—if it runs out, players lose the game immediately, forcing them to either use health-refilling items or race strategically to avoid damage.
- Leaderboards: Track players' time and showcases how quickly they reach the finish line. Players can compare their race times with others, adding a competitive element and encouraging players to improve their performance. The leaderboard provides a real-time ranking of players based on their race results.

THE GAME WORLD

Overview of Krash Kart: The Mess of Mazomski Invasions

In 2046, the world was hit by a sudden alien invasion. The Mazomski, a ruthless species with one giant eye and purple bodies, attacked cities and changed the skies, bringing chaos everywhere. They float in the air, terrifying all who see them.

Amid the destruction, individual racers, each in their own high-speed vehicles, compete to survive. They race across the ruined world, battling not only the Mazomski but also each other to gather resources and stay alive.

The Mazomski are not just invaders—they're also fierce competitors, sending their patrols, the Mazomski Military, to attack racers. To survive, you must be faster, smarter, and tougher than both the Mazomski and your rivals.

Krash Kart is a fast-paced racing game where two players race through dangerous tracks, face Mazomski enemies, and collect power-ups to stay ahead. The only rule? Survival of the fastest. Will you outrun the Mazomski and your competition, or crash and burn?

Game Engine

Overview

The game is developed in Unity 3D, creating a dynamic and realistic game world, which is custom-built to provide a unique gameplayer experience.

Lighting Setup

To create a well-lit and immersive environment, the game includes at least one ambient light, one directional light, and 1-2 point lights.

Screen

The game uses a split screen for 2 players, allowing each player to have their own view of the race.

Camera

Camera 1 follows Player 1, providing a clear view of their racing experience, while camera 2 follows Player 2, ensuring both players have an individual perspective while racing.

Scripting

The game uses C# for scripting, implementing a variety of AI and gameplay features.

Collision Detection

Unity’s OnTriggerEnter is used for collision detection, ensuring that interactions between game objects (such as power-ups and enemies) are properly handled.

Mechanism

Player

A simple input system is implemented to control the vehicle's movement. The system allows for basic controls such as moving forward, backward, left, and right, enabling the player to steer the car through the environment. Different acceleration and brake inputs are mapped to simulate realistic car behavior, with variable acceleration rates based on the player's input. Gravity is applied to the vehicle to ensure it reacts to the environment, such as maintaining proper traction on the ground or causing it to fall off edges or ramps realistically. The system provides an intuitive and responsive driving experience.

NPC

NPCs are powered by advanced AI systems to create dynamic and unpredictable gameplay.

Using a Finite State Machine (FSM), NPCs can be idle, patrol (tracking players), or attack when they detect players, all based on state change logic. Behavior trees manage more complex decision-making, enabling different behaviors within the same state, enhancing their overall intelligence and realism. Fuzzy logic allows for natural decision-making based on energy levels, factoring in three states—idle, patrol, and attack—along with strength differences among NPCs. This determines the level of attack or the speed at which they chase players.

Randomness and probabilities affect the locations of pickup items, making the game more challenging for players. They also introduce unpredictability into NPC behavior. Sensors enable NPCs to detect and respond to their environment, while path following and pathfinding allow them to navigate the world, react to obstacles, and approach players.

Additionally, flocking behavior is used for environmental effects, such as birds flying or clouds floating, adding realism to the world.

USER INTERFACE

Start Scene

- Start: Enter the Main Menu scene

Main Menu

- Play: Enter the game level
- Instructions: Show the instructions of the game
- Options: Show the audio and level options
- Exit: Exit the game

Instructions

- Player Name: Input field for players to input their names
- Controls: Show the control of each player
- Powerup Items: Show the available pick up items in the game and their functions
- Main Menu: Go back to the main menu

Options

- Audio: Opt in/ out the audio during the game
- Level: choose the game level, but the level will only be available after players finish the previous one
- Main Menu: Go back to the main menu

Game Scene

- Name: Display players' name
- Life: Players' remaining life (Increase when take reward items; Decrease when get attacks)
- Power up Showbox: Displays a grey icon when a power-up is picked up, highlights it during use, and removes it after use
- Timer: Real-time tracking players' race time

Game Over (Leaderboards)

- Display: Show race time of each player in different level
- Main Menu: Go back to the main menu
- Player Again: Enter the first level (easy level) of the game

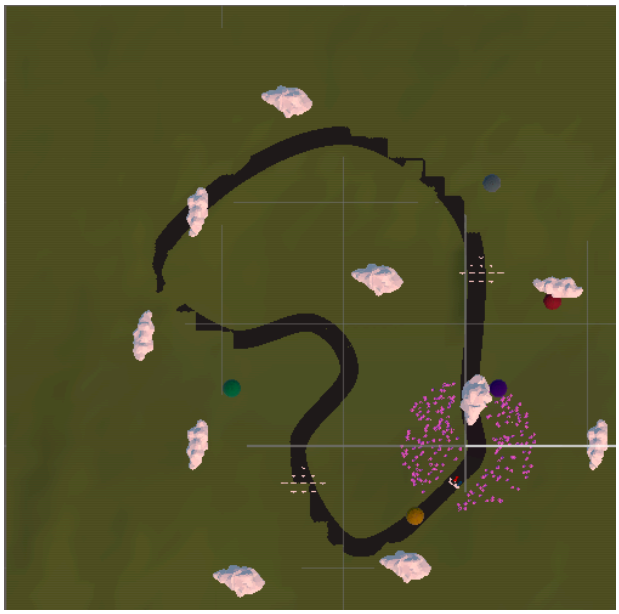
THE LEVEL LAYOUT

Easy Level

- A flat, simple racing track with minimal challenges.
- Randomly placed pickup items for the player to collect.
- Random NPC enemies with a mix of soft states, focusing more on passive behaviors (e.g., patrolling or idle).

Medium Level

- Steeper and more challenging racing tracks with varied terrain.
- Random pickup items, but with fewer opportunities compared to the easy level.
- Random NPC enemies with a mix of aggressive and soft states, making them more unpredictable and challenging.



Hard Level

- A difficult, challenging track with more obstacles and tighter corners.
- Fewer pickup items, making resource management more critical.
- NPC enemies become more aggressive, actively pursuing and attacking the player, making survival more difficult.

LEVEL EDITING

Terrain Editing

Using Unity's terrain tools, you can sculpt the terrain by raising and lowering the ground with heightmap adjustments, painting textures, and adding environmental features like roads, trees, balloons, buildings, and mountains.

Scripting and Logic

AI behavior, such as flocking (for environmental effects like birds flying or clouds floating), is implemented using Unity's scripting system (C#) to apply flocking algorithms or behaviors. For players and NPCs, their movement, behaviors, and state changes are controlled through C# scripting, using different logic, including FSM, behaviour tree, sensor, fuzzy logic, randomness and probabilities, to make the environment more dynamic and reactive to player actions.

Path Following and Pathfinding

C# Scripting in Unity is used to define paths or areas that NPCs and players can move through, and to test AI behavior. Path following and pathfinding algorithms are used to allow NPCs to follow these paths. This ensures that NPCs navigate the world realistically, avoiding obstacles and dynamically reacting to players and the environment.

Camera

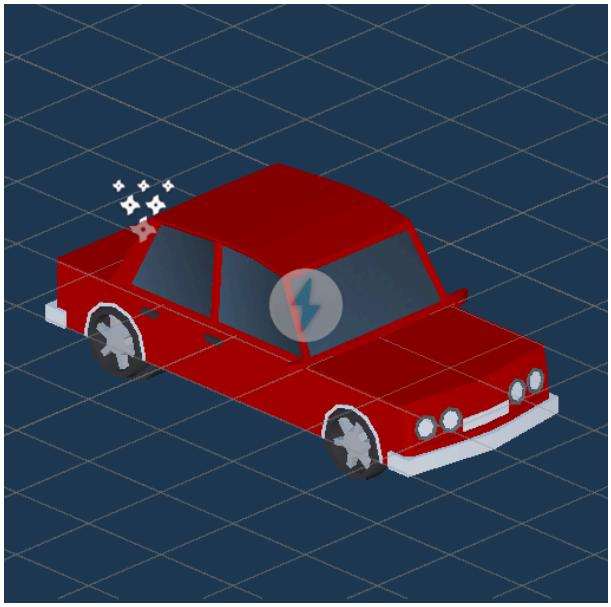
For two players, the screen is split into two sections with separate cameras following each player. Unity's camera system can be used to create split-screen functionality, with Camera 1 following Player 1 and Camera 2 following Player 2, ensuring each player has their own view of the action.

User Interface

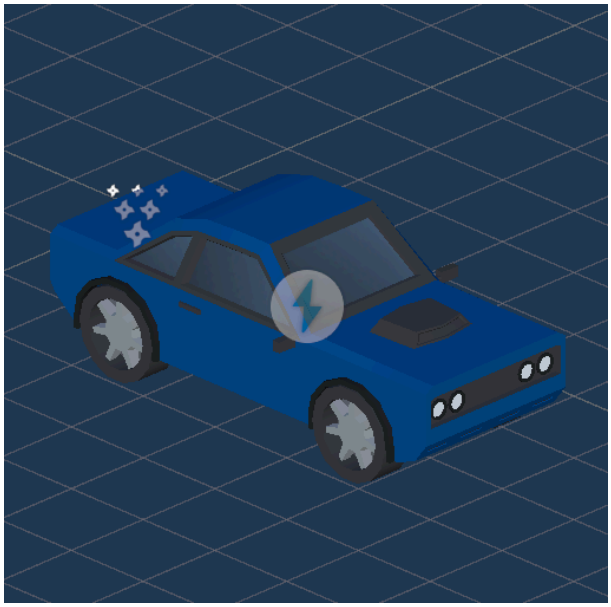
The UI will include buttons, dropdown lists, and input fields to allow players to interact with the game. Unity's UI system provides a flexible way to create interactive elements like start menus, options screens, and in-game HUDs. For example, buttons for starting the game, dropdown lists for selecting levels, input fields for entering names, and scene loading systems to move between different game levels.

GAME CHARACTERS

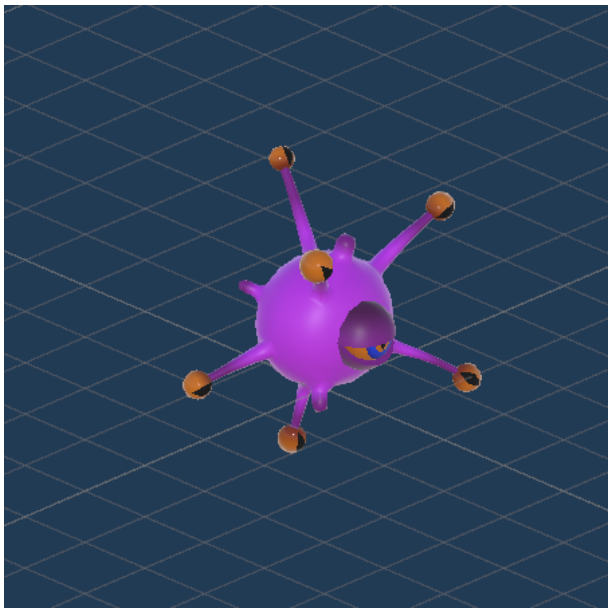
Player 1



Player 2



Mazomski (NPC)



CHARACTER MECHANISM

Players

Player Movement

- Input Handling: The player's movement is controlled through the Unity Input System, which maps player controls to specific actions (e.g., forward/backward and left/right).
- Acceleration/Deceleration: The player's speed is governed by the acceleration variable for forward movement, and brakeForce is applied when the player moves in reverse or stops.
- Speed Adjustment: The currentSpeed is clamped between -maxSpeed / 2 and maxSpeed, ensuring the player doesn't exceed the maximum speed or reverse beyond a certain limit.
- Turning: The steering is applied based on the input (left/right) and the player's current speed, with adjustments for smoother controls and reverse steering when moving backward.

Gravity and Grounding

- Vertical Movement: Gravity is applied to the player through the verticalVelocity variable. When the player is not grounded, gravity is added to pull the player down.
- Ground Check: A slight downward force (groundCheckOffset) ensures the player stays grounded, which is essential for proper physics handling.

Power-Up Handling

The player can pick up power-ups with OmTrigger Enter Collision that affect gameplay:

- Protected: Grants temporary protection (5.0f) from NPC attacks.
- Speed: Temporarily boosts the player's speed for a limited time (5.0f).

Sound Effect

A car sound is played when player movement is detected, and the sound stops when the player stops moving.

Transform

- Initiate Player Starting Position: Initiate Players start point in every level.
- Reset Player Position: Move players to a designated position when health under 0.

Input System Setup

- Different Controls for Player 1 and Player 2: The script distinguishes between Player 1 and Player 2 by assigning separate controls and actions to each player.
- Mapping of Actions: Input actions for movement and power-ups are mapped using Unity's Input System.

NPC

Decision-Making

This system effectively combines structured state management (FSM) with flexible decision-making (BT), resulting in NPCs that are highly adaptable, lifelike, and challenging for players to engage with. It also aligns with best practices in software design, making it robust and maintainable.

Finite State Machine (FSM)

Manage high-level states (Idle, Patrol, Attack) for NPCs.

- Used to decouple the creation of state objects from the main FSM logic, promoting modularity and scalability.
- Allows dynamic creation of state objects as NPCs transition between states, improving code reusability and enabling the addition of new states without modifying the core FSM logic.
- States:
    - Idle: Triggered when the energy level is low.
    - Patrol: Activated when the energy level is above a certain threshold.
    - Attack: Can be triggered by external stimuli (e.g., player proximity) and depends on additional conditions managed by the Behavior Tree.

Behaviour Tree (BT)

Add depth and decision-making logic within each high-level FSM state, creating variations in NPC behavior.

- Idle State:
    - High Idle Strength: Sensor remains inactive; the NPC ignores nearby players but can still take damage from collisions.
    - Low Idle Strength: Sensor activates, detecting nearby players and potentially transitioning the NPC to the Attack state.

- Patrol State:
    - High Patrol Strength: NPC follows predefined waypoints.
    - Low Patrol Strength: NPC stays in place but keeps its sensor active for player detection.
- Attack State:
    - NPC moves toward the player, with movement speed and attack damage scaled by attackStrength.

Movement

Sensor

Detect players within a specified area and influence NPC behavior.

- Sensors are typically implemented using Unity's colliders (e.g., Sphere Collider or Box Collider) set as triggers.
- When the player enters the sensor's area, the NPC evaluates its current state and strength levels to decide on an appropriate action:
    - Idle State: If the sensor is active (low idle strength), the NPC transitions to the Attack state upon detecting a player.
    - Patrol State: The sensor may act as an additional layer of vigilance. If triggered, the NPC may shift focus from patrolling to chasing the player.

Path Following

Define patrol behaviors by having NPCs move along randomized paths for diversity.

- Waypoints: The patrol path consists of waypoints, which are randomly selected for each NPC at runtime. This creates unique patrol routes for every NPC, adding variability.
- Movement: NPCs use path following algorithms to move smoothly between waypoints.
- Dynamic Adjustment
    - High patrol strength: The NPC actively follows the path, moving through multiple waypoints.
    - Low patrol strength: The NPC stays near its current waypoint or stops moving entirely.

Path Finding

Enable NPCs to navigate the environment and pursue players intelligently. If the player is detected by the sensor, the NPC transitions to the Attack State and uses pathfinding to chase the player.

- Movement: NPCs use path finding algorithms to ensure that the NPC avoids obstacles and calculates the shortest path to the player, implementing algorithms like A* or Dijkstra's algorithm to handle navigation.

- Dynamic Behavior:

    - High Attack Strength: NPC moves quickly and aggressively toward the player.
    - Low Attack Strength: NPC moves slower, possibly retreating if the player's proximity exceeds its comfort range.

Energy Level

- Randomized AnimationCurves for Behavior Strengths: Three AnimationCurve objects (idleCurve, patrolCurve, attackCurve) represent the strength profiles for idle, patrol, and attack behaviors. These curves are randomized at runtime using the RandomizeCurve method to ensure NPCs have unique behavior patterns.
- Energy Level Calculation: The energy level (energyLevel) is determined through the FuzzyLogic method, combining the strengths of different behaviors.
- Fuzzy logic rules ensure energy levels are adjusted dynamically based on various conditions: High energy favors patrol behaviors and low energy defaults to idle behaviors.

Transform

- Define a region using center point and radius calculation within the game world where NPCs can spawn.
- Within the spawn area, generate random positions using Unity's Random.Range for the X, Y, and Z coordinates.
- Ensure these positions are valid (e.g., not inside obstacles or out of bounds).

Other NPCs

Pick-up Items

- Define a region using center point and radius calculation within the game world where NPCs can spawn.
- Within the spawn area, generate random positions using Unity's Random.Range for the X, Y, and Z coordinates.
- Ensure these positions are valid (e.g., not inside obstacles or out of bounds).

Environment

Using flocking behavior, which involves three primary rules: Separation, Alignment, and Cohesion. These rules enhance the game world by creating more natural and dynamic movement, such as simulating birds flying together or clouds drifting in the sky.

PICKUP ITEMS

Power Up Items

- Randomly positioned in the game level
- Players collide to pick it up
- Randomly earn the power up, either Speed Up (60%) or Protect (40%)

Speed Up

- Players get 5 times speed up for 5 seconds

Protect

- Players get protected from attacks for 5 seconds

Reward Items

- Randomly positioned in the game level
- Players collide to earn life +10

SCORING SYSTEM

Health

The Health System tracks the player's health, which decreases as they take damage from NPC attacks. When a player's health drops below 0, they are immediately transformed to the finish line, effectively "resetting" their position in the race.

- Default Health Level: Players start with 100 health.
- Health Reward: Players can gain +10 health as a reward in certain conditions (e.g., pickups or achievements).
- NPC Attack: NPCs deal damage based on their attack strength level, which influences the amount of health lost when hit.

Timer

The **Timer** system tracks each player's race time in real-time, updating continuously as they progress. At the game over scene, players' race times for each level are displayed, adding a competitive element and allowing for performance comparisons.

Assets Acquired

Players

Low Poly Free Vehicles Pack by Palmov Island
https://assetstore.unity.com/packages/3d/vehicles/low-poly-free-vehicles-pack-228368?srsltid=AfmBOor-HRYRy1A-CxsV5PojrNDBBSDUOWk1g24KQHq76n_YFxHTSDSZ

NPC Enemies (Mazomski)

RPG Monster Partners PBR Polyart by Dungeon Mason
https://assetstore.unity.com/packages/3d/characters/creatures/rpg-monster-partners-pbr-polyart-168251?srsltid=AfmBOoomxmCdiUtdDF-b5gKpBteMUOdroniv9wIVoOHFvloXOW7MWpLp

Environment

Cartoon Race Track - Ovel by RCC Design
https://assetstore.unity.com/packages/3d/environments/roadways/cartoon-race-track-oval-175061

EasyRoads3D Free v3 by AndasOFT
https://assetstore.unity.com/packages/3d/characters/easyroads3d-free-v3-987

Fence Layout Tool by PolygonPi
https://assetstore.unity.com/packages/tools/utilities/fence-layout-tool-162856

LOW POLY - Magical Potions Pack by Agwyn Studios
https://assetstore.unity.com/packages/3d/props/low-poly-magical-potions-pack-227572?srsltid=AfmBOooqNZCBrtgpW2UbogoaXv2dIkUZIAD-tKuBpzHxOvVh1nFhIe7tk

Low Poly Environment Park by Palmov Island
https://assetstore.unity.com/packages/3d/environments/low-poly-environment-park-242702

RPG Poly Pack - Lite by Gigel
https://assetstore.unity.com/packages/3d/environments/landscapes/rpg-poly-pack-lite-148410?srsltid=AfmBOor5YEH6zYRCaI13mCDUaMynYA3TbMJqoA4mUmbXKL3G8FbanDjL

Simplistic Low Poly Nature by Acorn Bringer
https://assetstore.unity.com/packages/3d/environments/simplistic-low-poly-nature-93894

GUI

Space Game Icon Set by FlexUnit
https://assetstore.unity.com/packages/2d/gui/icons/space-game-icon-set-290437

Game Input Controller Icons Free by Amanz
https://assetstore.unity.com/packages/2d/gui/icons/game-input-controller-icons-free-285953

MUSICAL AND SOUND EFFECTS

Music

- Background music: Racing Trap (Action Speed Background Sport Music Intro Theme) by MFCC
- Source: https://pixabay.com/music/trap-racing-trap-action-speed-background-sport-music-intro-theme-115039/

Sound

- Car sound effect: car changing gears sound by aenjoytrips
- Source: https://pixabay.com/users/aenjoytrips-32262352/