

overtaker-car-racing-ue5-game-dev

⚡ Plug & Play Edition — Just unzip and play!

💾 Download Game the Playable Build for Windows:

 Download ZIP

⌚ Release:

Gameplay:

Overtaker is a vibrant, arcade-style racing game built in **Unreal Engine 5.6**, featuring AI-powered opponents and dynamic track-following mechanics. The project combines stylized visuals with responsive vehicle physics, offering a complete racing experience with checkpoints, lap timing, and competitive AI racers that navigate custom spline-based tracks—all wrapped in a colorful, Mario Kart-inspired aesthetic.

 alt text

- Author: [Kintsugi-Programmer](#)

Disclaimer: The content presented here is a curated blend of my personal learning journey, experiences, open-source documentation, and invaluable knowledge gained from diverse sources. I do not claim sole ownership over all the material; this is a community-driven effort to learn, share, and grow together.

Controls

| Action | Keyboard | Controller |
|--------|----------|------------|
|--------|----------|------------|

- ✓

All assets organized in folders:

- Levels
- Materials
- Sounds
- Generated (modeling assets)
- Megascans (textures and decals)

MIT License

Copyright (c) 2026 Siddhant Bali

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Game Design & Development Documentation

Table of Contents

- [overtaker-car-racing-ue5-game-dev](#)
- Game Design & Development Documentation
 - Table of Contents
 - Project Setup
 - Initial Configuration
 - Project Structure
 - Enable Git LFS
 - AI Car Implementation
 - Creating AI Car Blueprint
 - AI Movement Setup
 - Path Following System
 - AI Steering Logic Function
 - Event Graph Integration
 - Vehicle Speed Adjustments
 - AI Car Settings
 - Player Car Settings
 - UI Timer Adjustment
 - Extending Countdown Timer
 - Car Color Customization
 - Creating Color Materials
 - Random Color System (AI Cars)
 - Manual Color Assignment (Alternative)
 - Player Car Color
 - Level Creation - Custom Race Track
 - Creating New Map
 - Landscape Spline Track Creation
 - World Settings Configuration
 - Player Start Placement
 - World Partition Fix
 - Material and Environment Setup
 - Importing Assets via Fab Marketplace

- Applying Grass Material
- Applying Road Material
- Placing AI Cars and Spline
 - AI Car Placement
 - Track Spline Setup for New Map
- Checkpoint System
 - Required Components from Template
 - Copying to New Map
 - Manual Checkpoint Placement
- Camera Adjustment
 - Player Camera Settings
- Environmental Polish
 - Fog System Enhancement
 - Foliage System - Trees
 - Foliage System - Rocks and Props
- Project Download and Resources
 - Unreal Club Membership
- Final Game Features Summary
 - Completed Systems
 - Future Customization Options
- Critical Technical Notes
 - Browser Storage Restriction
 - Common Issues and Solutions

Project Setup

Initial Configuration

- **Engine Version:** Unreal Engine 5.6 (recommended or above)
- **Template:** Vehicle Template (Games section)
- **Blueprint Type:** Blueprint (not C++)
- **Variant:** Time Trial (provides timer and race features)

Project Structure

The template provides:

- A basic drivable car with responsive controls
- A variant map with complete racetrack
- Timer system and checkpoints
- Foundation for racing mechanics

Enable Git LFS

- Git LFS stores large binary assets (UE5 .uasset, .umap, textures, audio, video, etc.) outside normal Git history to keep the repo small and fast.

```
# Install Git LFS

# Windows (PowerShell / CMD)
winget install Git.GitLFS          # Installs Git LFS on Windows

# macOS (Homebrew)
brew install git-lfs               # Installs Git LFS using Homebrew

# Linux (Debian/Ubuntu)
curl -s https://packagecloud.io/install/repositories/github/git-lfs/script.deb.sh | sudo bash
sudo apt-get install git-lfs       # Installs Git LFS package

# Initialize Git LFS in the current repository
git lfs install                   # Enables Git LFS for this repo

# Track Unreal Engine and other large binary assets
git lfs track "*.*asset"          # Unreal Engine asset files
git lfs track "*.*umap"            # Unreal Engine map files
git lfs track "*.*fbx"             # 3D models
git lfs track "*.*wav"              # Audio files
git lfs track "*.*mp4"              # Video files
git lfs track "*.*png"              # Texture/images

# Add LFS configuration file
git add .gitattributes             # Adds LFS tracking rules

# Add remaining project files
git add .                           # Stages all project files

# Commit the changes
git commit -m "Enable Git LFS and track binary asset types"

# Push changes to GitHub
git push origin main                # Pushes commits to the main branch
```

AI Car Implementation

Creating AI Car Blueprint

Step 1: Duplicate Base Car

```
Location: Content/VehicleTemplate/Blueprints/
- Find: BP_SportsCar (child class, actual spawned vehicle)
- Duplicate it
- Rename: BP_Car_AI
```

Note: BP_SportsCar inherits from VehicleBase_Pawn (parent class containing main logic for brakes, throttle, etc.)

AI Movement Setup

In BP_Car_AI Event Graph:

1. Begin Play Node - Initial setup

- Add: Begin Play event
- Purpose: Set constant throttle for AI

2. Set Throttle Input

Component: Vehicle Movement Component
Node: Set Throttle Input
Value: 0.5 (range 0-1, can exceed 1 for higher speed)

3. Set Steering Input (temporary test)

Node: Set Steering Input
Test Value: 1 (full right)
Note: This is just for testing initial movement

Path Following System

Creating Track Spline Blueprint:

Location: Content/Blueprints/ (new folder)

Type: Actor Blueprint

Name: BP_TrackSpline

Components:

- Add: Spline Component
- No additional logic needed in Event Graph

Spline Placement in Level:

- Drag BP_TrackSpline into level
- Click ending point to move/adjust
- Hold **Alt** to create new points
- Press **End** key after each point to snap to ground (CRITICAL)
- Continue until track path is complete
- Deselected view may show spline through road (visual only, actual position is correct)

AI Steering Logic Function

Create Function: "Follow Track Spline"

Input Parameter:

Name: Spline Track
Type: BP_TrackSpline (Object Reference)

Function Logic:

1. Get Spline Tangent Direction

From: Spline Track → Get Spline Component
Node: Find Tangent Closest To World Location
Coordinate Space: World (not Local - IMPORTANT)
World Location: Get Actor Location

2. Normalize and Calculate

→ Normalize (keeps range 0-1)
→ Multiply by 500 (convert to Float)
→ Add with Get Actor Location

3. Find Target Location

Spline Component → Find Location Closest To World Location
Input: Previous calculation result
Coordinate Space: World

4. Calculate Rotation

Node: Find Look At Rotation
Start: Get Actor Location
Target: Result from Find Location
→ Split Struct (Z axis only needed for left/right steering)

5. CRITICAL FIX - Delta Rotator

Node: Delta Rotator (normalizes rotation between 0-1)
A: Look At Rotation (Z split)
B: Get Actor Rotation
→ Split result, use Z value

6. Map to Steering Range

```
Node: Map Range Clamped  
Value: Delta Rotator Z result  
In Range A: -90  
In Range B: 90  
Out Range A: -1  
Out Range B: 1  
Purpose: Convert -90/90 degrees to -1/1 steering input
```

7. Output

```
Type: Float  
Name: Steering Value
```

Event Graph Integration

Event Tick Implementation:

```
Event Tick (executes every frame)  
→ Follow Track Spline  
    Input: Track Spline variable  
→ Get result (Steering Value)  
→ Vehicle Movement Component → Set Steering Input
```

Begin Play - Spline Reference:

```
Begin Play  
→ Get Actor of Class (singular, not plural)  
    Class: BP_TrackSpline  
→ Promote to Variable: "Track Spline" or "BP_TrackSpline"  
→ Use in Event Tick
```

Throttle Note:

- Throttle remains at 0.5 from Begin Play (Set once, stays constant)
- Steering updates every frame via Event Tick (needs continuous updates)

Vehicle Speed Adjustments

AI Car Settings

Location: BP_Car_AI
Component: Vehicle Movement Component
Settings:
- Max Torque: 500 (down from default)
- Max RPM: 3500 (down from default)
Purpose: Slower speed for better track following

Player Car Settings

Location: VehicleTemplate/Blueprints/VehicleAdvancedSportsCar
Component: Vehicle Movement Component
Settings:
- Max Torque: 500 (match AI)
- Max RPM: 3500 (match AI)
Purpose: Fair competition

UI Timer Adjustment

Extending Countdown Timer

Location: Variant Template/UI/StartWidget

In Event Graph (Event Construct):

Original countdown: 3 seconds (3, 2, 1, GO) New countdown: 5 seconds (5, 4, 3, 2, 1, GO)

Changes:

1. Initial timer value: Change from 3 to 5
2. Multigate pins: Add 2 additional output pins
3. New outputs:
 - o Pin 0: "5" (NEW)
 - o Pin 1: "4" (NEW)
 - o Pin 2: "3" (was Pin 0)
 - o Pin 3: "2" (was Pin 1)
 - o Pin 4: "1" (was Pin 2)
 - o Pin 5: "GO" (unchanged)
4. Connect each pin to Play Animation node
5. Purpose: Give AI more time to accelerate fairly

Car Color Customization

Creating Color Materials

Location: Navigate to sports car body material

Process:

1. Find: Sports Car Body material

2. Duplicate and rename:

- `MaterialName_Blue`
- `MaterialName_Green`
- `MaterialName_Purple`
- Keep original for orange/default

3. Edit each material:

- Open material
- Modify Base Color to desired color
- Save

Random Color System (AI Cars)

In BP_Car_AI Construction Script:

Construction Script executes in editor (not just during play)

1. Get: Chassis Main (main car body mesh)
2. Node: Set Material
Element Index: 1 (not 0 - chassis uses slot 1)
3. Create Variable: "Colors" (Material array type)
 - Change from single to Array
 - Add materials:
 - * Purple material
 - * Green material
 - * Orange material
 - * (NOT blue - reserved for player)
4. Logic:
Get Colors array
→ Random Array Item
→ Set Material

Result: Each AI car gets random color from list on spawn

Manual Color Assignment (Alternative)

Instead of random system:

1. Disconnect random color logic
2. Place AI cars in level
3. Select each car individually
4. Find: Chassis Main component
5. Manually set Material slot 1 to desired color
6. Ensures no duplicate colors in race

Player Car Color

Location: BP_VehicleAdvancedSportsCar

Component: Chassis Main
Material Slot 1: Blue material
Purpose: Distinct from AI cars

Level Creation - Custom Race Track

Creating New Map

File → New Level → Open World template
Purpose: Provides hills/mountains for interesting track layout

Landscape Spline Track Creation

Step 1: Switch to Landscape Mode

Selection Mode → Landscape Mode

Step 2: Enable Splines

Manage tab → Click "Splines"

Step 3: Create Layer

Layers section:
- Add new layer
- Enable "Layer Splines"
- Select the new layer (IMPORTANT)
- Return to Splines section

Step 4: Create Track Path

Controls:
- Hold Command (Mac) or Control (Windows) + Left Click to add points
- Click on starting point at end to close loop
- Track automatically carves into terrain

Step 5: Add Road Mesh

1. Click: Update Spline Mesh Levels (selects all spline points)
 2. Scroll to: Spline Meshes section
 3. Add Element
 4. Search: "track"
 5. Select: SM_Track_10M (included in race template)
- Result: Road mesh fills spline path

World Settings Configuration

Window → World Settings (if not visible)

Settings:

- Game Mode Override: Time Trial Game Mode
(provides timer and race functionality)

Player Start Placement

1. Quickly Add → Player Start
2. Place at desired starting position on track
3. Press End key to snap to ground

Note: Delete any existing Player Start actors in level

World Partition Fix

Issue: Map appears black on first play (World Partition system)

Solution:

File → Recent → Reopen current map
This fixes World Partition loading

Disable World Partition (for this project):

World Settings → Search "World Partition"
→ Disable "Enable Streaming"
→ Click Yes
→ Save and reload level
Purpose: Ensures entire map renders (including distant mountains)

Material and Environment Setup

Importing Assets via Fab Marketplace

Grass Material:

1. Window → Fab (Unreal Marketplace)
2. Login to account
3. Search: "stylized grass"
4. Filter: Materials and Textures category
5. Filter: Price → Free
6. Find: "Pack Bonus Textures"
7. Add to Project
8. Wait for import

Road Material:

Search: "road"
Filter: Free
Find: "Asphalt Road" by Quickso
Add to Project (version 5.4 compatible with 5.6)

Applying Grass Material

Select Landscape:

- Select PARENT landscape actor (not sections)
- Critical: Must be parent, not individual landscape sections

Apply Material:

Details Panel:
- Scroll to: Landscape Material
- Drag grass material into field

Adjust Tiling:

Issue: Material appears too small/detailed
Fix:

1. Double-click material to open
2. Find: TextureCoordinate node
3. Change values:
 - U: 0.1
 - V: 0.1
4. Apply and Save

Result: Larger, more visible pattern (Mario Kart style)

Applying Road Material

Location: Find track mesh (SM_Track_10M)

1. Open static mesh
2. Materials section shows 2 materials:
 - Slot 0: Edge (keep default)
 - Slot 1: Road surface (replace)
3. Navigate to road material instance
4. Apply to Slot 1

Customize Road Color:

1. Open road material
2. Adjust Base Color to darker shade (more realistic asphalt)
3. Apply and Save

Placing AI Cars and Spline

AI Car Placement

Location: VehicleTemplate/Blueprints/SportsCar/BP_Car_AI

1. Drag AI car into level at starting line
2. Widget size issue fix:
 - Temporarily reduce UI scale for easier camera control
 - Increase again after placement
3. Duplicate cars:
 - Select car
 - Transform mode: Local
 - Hold Alt + drag to duplicate
 - Press End to snap to ground
 - Repeat for desired number of AI cars
4. Assign colors:
 - Select each car
 - Component: Chassis Main
 - Set material to unique color (purple, green, orange)
 - Player car remains blue

Track Spline Setup for New Map

Location: Content/Blueprints/BP_TrackSpline

1. Drag into level at starting position
2. Select first point, move to front of first car
3. Press End to snap to ground
4. Hold Alt + drag to duplicate points
5. Press End after each point (CRITICAL for accuracy)

Important Viewing Tips:

- Don't view from single angle only
- Use top-down view primarily for accuracy
- Check from multiple angles to avoid misplacement
- Enable Unlit view mode for better spline visibility:

Viewport → Unlit mode

Result: No lighting, easier to see spline against track

Spline Creation Process:

- Manual process, takes time
- Follow track path closely
- Regular points on straights
- More points on curves for accuracy
- Complete entire track loop

Checkpoint System

Required Components from Template

From Time Trial Template Level:

Track Gates (Checkpoints):

- Location: Scattered along track
- Purpose: Prevent shortcutting, ensure full track completion
- Blueprint: Track Gate actor

Finish Line:

- Location: End of track (also start in loop)
- Purpose: Race completion trigger
- Blueprint: Finish Line actor
- Settings: "Is Finish Line" enabled

Copying to New Map

1. Open Time Trial template level
2. Select all Track Gates and Finish Line
3. Copy (Ctrl+C)
4. Return to race map
5. Paste (Ctrl+V) - optional, or place manually

Manual Checkpoint Placement

Track Gate Placement:

1. Locate Track Gate blueprint
2. Drag into level at strategic points:
 - Corners (prevent cutting)
 - Long straights (ensure progression)
 - Before finish line
3. Rotate to span track width
4. Scale to cover full road

Linking Checkpoints:

CRITICAL PROCESS for each checkpoint:

1. Select PREVIOUS checkpoint
2. Details → Next Marker
3. Click eyedropper/pipette icon
4. Click on NEXT checkpoint in viewport
5. This creates progression chain

Example chain:

Start → Gate1 → Gate2 → Gate3 → Finish
Each points to next in sequence

Finish Line Setup:

1. Place at end of track (start position for loop)
2. Scale to cover track width
3. Settings: Check "Is Finish Line"
4. Link from last Track Gate:
 - Select last Track Gate
 - Next Marker → Select Finish Line
5. Finish Line has NO next marker (end of chain)

Hide in Game:

All Track Gates and Finish Line:
- Details → Search "Hidden in Game"
- Enable checkbox
Result: Visible in editor, invisible during gameplay

Camera Adjustment

Player Camera Settings

Location: BP_VehicleAdvancedSportsCar (player blueprint)

Component: Spring Arm (camera attachment)

Adjustable Settings:

1. Socket Offset (Z-axis)

Controls: Height above car
Recommended: 100
Range to test: 50-150

2. Arm Length

Controls: Distance behind car
Recommended: 450
Range to test: 350-500

Final Recommended Values:

Socket Offset Z: 100
Arm Length: 450
Result: Better forward visibility, can see track ahead

Testing Process:

- Adjust values
- Compile and save
- Press Play to test
- Iterate until comfortable view achieved
- Avoid: Too high (toy-like), too close (can't see ahead), too far (lose detail)

Environmental Polish

Fog System Enhancement

Component: Exponential Height Fog (in level)

Settings to Modify:

1. Enable Volumetric Fog

Default: Disabled
Change: Enable checkbox
Effect: Adds atmospheric depth

2. Fog Density

Setting: Move to maximum
Effect: Thicker fog coverage

3. Fog Falloff

Action: Decrease value
Effect: More pronounced foggy atmosphere

4. Second Fog Data

Option: Enable and adjust
Effect: Additional fog layer for complexity

5. Volumetric Fog Settings

Extinction Scale: Controls fog thickness in air
Scatter Distance: Adjusts volumetric quality
Start Distance: Where fog begins

Goal: Cinematic, atmospheric look

Result: Hides empty distant areas, adds depth, more professional appearance

Foliage System - Trees

Import Tree Assets:

Window → Fab
Search: "stylized environment pack"
Filter: Free

```
Select: Stylized Environment Pack  
Add to Project (version 5.4, compatible with 5.6)  
Wait for import
```

Foliage Mode Setup:

Selection Mode → Foliage Mode

1. Navigate: Content/StylizedEnvironment/Meshes
2. Find tree mesh
3. Drag tree into Foliage dropdown
4. Automatic prompt: Create foliage type asset
5. Click Save (automatic asset creation)

Foliage Settings:

Select tree in foliage list:

- Enable checkbox (to paint)

Density: 10 (prevents overcrowding)

Scale Settings:

- Minimum: 1.5
- Maximum: 1.8
- Random: Enabled (for variety)

Brush Size: Adjust to preference (smaller = more control)

Painting Trees:

Performance Warning: Don't overpaint (causes lag)

Strategy:

- Paint along track edges (creates depth illusion)
- Focus on corners (most visible during racing)
- Add sparse distant trees (background layer)
- Avoid center of track area

Wind Animation Adjustment:

Issue: Trees have excessive wind movement

Fix - Trunk Material:

1. Open tree static mesh
2. Materials section → Tree trunk material
3. Open material editor
4. Find: World Position Offset nodes
5. Multiply node values:
 - Change: 0.5 → 0.2 (or 0.1)
6. Apply and Save

Fix - Leaves Material (Material Instance):

1. Select leaves material
2. Parameters:
 - Wind Intensity: 0.1
 - Wind Speed: 0.1
3. Save

Result: Subtle, realistic wind movement

Foliage System - Rocks and Props

Adding Rock Foliage:

From Stylized Environment Pack:

1. Find rock meshes
2. Drag ONE rock type into foliage list
3. Settings:
 - Density: 10 (or 5-8 for variety)
 - Scale: 5 (min) to 8 (max)
 - Random scale: Enabled
4. Disable tree painting (uncheck tree)
5. Paint rocks along track borders

Adding Wooden Logs:

1. Find log meshes in pack
2. Add to foliage list
3. Reduce density (prevents overcrowding)
4. Scale: 5-8 range
5. Disable other foliage types
6. Paint strategically along track

Performance Note:

Foliage meshes (rocks, logs):
- Enable Nanite (if available)
- Better performance than traditional foliage
- Meshes handle Nanite better than grass/leaves

Painting Strategy:

Layer Philosophy:

1. Trees: Main depth layer
2. Rocks: Medium detail layer
3. Logs: Accent/variation layer

Balance: Visual interest vs. performance

View angles: Looks better in motion during racing

Project Download and Resources

Unreal Club Membership

Included Content:

- Complete racing game project file
- 200+ tutorial assets
- Pre-made blueprints and systems
- Mechanics from viral games (featured in streamer content)
- Migration-ready assets for own projects
- This tutorial's complete project

Access: Link in video description

Final Game Features Summary

Completed Systems

1. AI Racing Cars

- Autonomous path following via splines
- Dynamic steering calculations
- Balanced speed with player

2. Race Track

- Custom landscape with road mesh
- Checkpoint validation system
- Finish line detection
- Loop-capable track design

3. Visual Polish

- Stylized grass terrain material
- Custom road surface
- Atmospheric fog system
- Foliage layers (trees, rocks, logs)
- Car color variety

4. UI Systems

- Extended countdown timer (5 seconds)
- Checkpoint tracking
- Race completion detection

5. Camera System

- Optimized viewing angle
- Better forward visibility

6. Environmental Details

- Wind-animated trees
- Varied rock placement
- Decorative props
- Hidden checkpoint markers

Future Customization Options

Suggested Enhancements:

- Custom car models
- Additional environment assets
- More sophisticated AI behaviors
- Power-up systems (Mario Kart style)
- Lap counting system
- Position tracking
- Multiplayer support
- Additional race tracks
- Weather effects
- Particle effects for impacts
- Sound effects and music

Critical Technical Notes

Browser Storage Restriction

NEVER use in artifacts:

- localStorage
- sessionStorage
- Any browser storage APIs

Reason: Not supported in Claude.ai environment, causes artifact failure

Alternatives:

- React state (useState, useReducer) for React components
- JavaScript variables/objects for HTML artifacts
- All data stored in memory during session

Common Issues and Solutions**AI Car Going Wrong Direction:**

- Missing Delta Rotator node in steering calculation
- Coordinate space set to Local instead of World
- Spline points not snapped to ground

Black Screen on Map Load:

- World Partition enabled (disable it)
- Need to reload map after opening

Checkpoints Not Working:

- Next Marker not linked properly
- Missing pipeline connection between gates
- Finish Line not marked as finish

Performance Issues:

- Too much foliage painted
- Nanite not enabled on static meshes
- World Partition disabled when needed for large maps

Camera Issues:

- Spring arm length too short/long
- Socket offset height incorrect
- Need to compile and save before testing

End-of-File

The [KintsugiStack](#) repository, authored by Kintsugi-Programmer, is less a comprehensive resource and more an Artifact of Continuous Research and Deep Inquiry into Computer Science and Software Engineering. It serves as a transparent ledger of the author's relentless pursuit of mastery, from the foundational algorithms to modern full-stack implementation.

Made with ❤️ [Kintsugi-Programmer](#)