Racing Car Physics - Master Implementation Plan

Milestones

- [x] **Milestone 1: Foundation (ME09)** - COMPLETED
  - [x] Suspension physics system
  - [x] Visual body tilt and roll
  - [x] Performance optimization for mobile
- [ ] **Milestone 2: Camera & Environment (ME05, ME06)** - IN PROGRESS
  - [ ] Incline/decline camera effects
  - [ ] Environment collision refinement
  - [ ] Integration testing
- [ ] **Milestone 3: Car Collision (ME07)** - PLANNED
  - [ ] Collision channel setup
  - [ ] Kinematic fake body implementation
  - [ ] Overlap detection and visual effects
- [ ] **Milestone 4: Airborne Mechanics (ME08)** - IN PROGRESS
  - [ ] Ramp boost system
  - [ ] Air control refinement
  - [ ] Auto-rotate and landing

Task Breakdown

Phase 1: ME05 - Incline & Decline Camera (3-5 days)

Task 1.1: Incline Detection System

Estimated Effort: 1 day
Priority: High
Dependencies: None

- [ ] Implement road normal raycast detection
  - [ ] Add raycast from car to ground
  - [ ] Calculate incline angle from normal
  - [ ] Determine incline vs decline direction
- [ ] Create FInclineData structure
  - [ ] Store incline angle
  - [ ] Store incline/decline state
  - [ ] Store acceleration multiplier

- [ ] Add update frequency optimization
    - [ ] Update detection every 0.1s (not every frame)
    - [ ] Cache results between updates

Acceptance Criteria:

- Incline angle accuracy within ±1 degree
- Detection updates at 10 Hz
- No performance impact on mobile (<0.05ms per update)

Task 1.2: Camera FOV Adjustment

Estimated Effort: 1 day
Priority: High
Dependencies: Task 1.1

- [ ] Extend AFollowCarCamera with incline settings
    - [ ] Add FInclineCameraSettings structure
    - [ ] Add FOV multiplier properties (1.1x for incline/decline)
    - [ ] Add interpolation speed property
- [ ] Implement UpdateCameraOnIncline function
    - [ ] Calculate target FOV based on incline state
    - [ ] Smooth interpolation to target FOV
    - [ ] Apply to camera component
- [ ] Test FOV transitions
    - [ ] Smooth entry to incline
    - [ ] Smooth exit from incline
    - [ ] No jarring transitions

Acceptance Criteria:

- FOV boosts to 110% on incline/decline
- Interpolation completes in <0.5 seconds
- No motion sickness reported in testing

Task 1.3: Camera Position Offset

Estimated Effort: 1 day
Priority: Medium
Dependencies: Task 1.2

- [ ] Add position offset properties
    - [ ] Z-axis offset (+50cm)
    - [ ] X-axis offset (-30cm backward)

- [ ] Implement offset application
    - [ ] Calculate target offset based on incline state
    - [ ] Smooth interpolation to target offset
    - [ ] Apply to spring arm socket offset
- [ ] Test for performance impact and lag only
    - [ ] Verify no frame rate drops
    - [ ] Ensure smooth interpolation

Acceptance Criteria:

- Camera raises 50cm on incline
- Camera moves back 30cm on incline
- Smooth transitions without jitter
- No performance impact or lag detected

Task 1.4: Acceleration Bonus

Estimated Effort: 0.5 day
Priority: Medium
Dependencies: Task 1.1

- [ ] Create FInclineAccelerationSettings
    - [ ] Uphill multiplier: 1.3x
    - [ ] Downhill multiplier: 1.1x
    - [ ] Flat multiplier: 1.0x
- [ ] Implement GetAccelerationMultiplier
    - [ ] Return multiplier based on incline state
- [ ] Apply to throttle input
    - [ ] Multiply engine force by multiplier

Acceptance Criteria:

- 130% acceleration on uphill
- 110% acceleration on downhill
- Feels rewarding to players

Task 1.5: Testing & Polish

Estimated Effort: 0.5 day
Priority: High
Dependencies: Tasks 1.1-1.4

- [ ] Unit tests

    - [ ] Incline angle calculation
    - [ ] FOV adjustment
    - [ ] Position offset
    - [ ] Acceleration multiplier

- [ ] Integration tests
    - [ ] Works with suspension system
    - [ ] No conflicts with other camera effects

- [ ] Manual testing
    - [ ] Feels natural on various slopes
    - [ ] No motion sickness
    - [ ] Performance on Android/iOS

Acceptance Criteria:

- All unit tests pass
- 30 FPS on Android, 60 FPS on iOS
- Positive playtest feedback

Phase 2: ME06 - Environment Collision Refinement (2-3 days)

Task 2.1: Angle Reduction Implementation

Estimated Effort: 1 day
Priority: High
Dependencies: None

- [ ] Create FCollisionCorrectionSettings
    - [ ] Angle reduction factor: 0.5 (50%)
    - [ ] Interpolation speed: 3.0
    - [ ] Max correction torque

- [ ] Implement CalculateTargetDirection
    - [ ] Get racing line direction
    - [ ] Calculate current angle to wall
    - [ ] Reduce angle by 50%
    - [ ] Return target direction

- [ ] Update ApplyCorrectionTorque
    - [ ] Use smooth interpolation
    - [ ] Apply reduced angle correction

Acceptance Criteria:

- Collision angle reduced by 50% ± 10%

- Smooth interpolation (no jerk)
- Feels helpful, not intrusive

Task 2.2: Collision Severity System

Estimated Effort: 0.5 day
Priority: Medium
Dependencies: Task 2.1

- [ ] Create ECollisionSeverity enum
  - [ ] Small (<15°)
  - [ ] Medium (15-45°)
  - [ ] Large (>45°)
- [ ] Implement ClassifyCollisionSeverity
  - [ ] Calculate collision angle
  - [ ] Return severity classification
- [ ] Implement GetCorrectionStrength
  - [ ] Small: 0.3 (gentle)
  - [ ] Medium: 0.6 (moderate)
  - [ ] Large: 1.0 (strong)

Acceptance Criteria:

- Severity classification accurate
- Correction strength scales appropriately
- Large angles get stronger correction

Task 2.3: Wall-Stick Prevention

Estimated Effort: 0.5 day
Priority: High
Dependencies: Task 2.2

- [ ] Implement PreventWallStick
  - [ ] Detect small angle collisions
  - [ ] Apply gentle outward impulse
  - [ ] Temporarily reduce friction
  - [ ] Reset friction after 0.2s
- [ ] Create low friction physics material
  - [ ] Friction: 0.1
  - [ ] Restitution: 0.5

Acceptance Criteria:

- No wall-stick at small angles
- Car separates from wall smoothly
- Friction resets correctly

Task 2.4: Testing & Polish

Estimated Effort: 1 day
Priority: High
Dependencies: Tasks 2.1-2.3

- ☐ Unit tests
  - ☐ Angle reduction calculation
  - ☐ Severity classification
  - ☐ Wall-stick prevention
- ☐ Integration tests
  - ☐ Works with racing line system
  - ☐ Various wall angles (0-90°)
- ☐ Manual testing
  - ☐ Feels forgiving
  - ☐ Maintains flow
  - ☐ Performance on mobile

Acceptance Criteria:

- All tests pass
- Positive playtest feedback
- No performance regression

Phase 3: ME07 - Car Collision Implementation (5-7 days)

Task 3.1: Collision Channel Setup

Estimated Effort: 0.5 day
Priority: Critical
Dependencies: None

- ☐ Create CarPriority collision channel
  - ☐ Edit → Project Settings → Collision
  - ☐ New Object Channel: CarPriority
  - ☐ Default Response: Block
- ☐ Create collision presets
  - ☐ PlayerCarPreset (CarPriority channel)
  - ☐ FakeBodyPreset (WorldDynamic channel)

    - [ ] OverlapBoxPreset (CarPriority, query only)

- [ ] Document collision matrix

    - [ ] Player car vs environment: Block
    - [ ] Player car vs AI: Ignore
    - [ ] Fake body vs AI: Block

Acceptance Criteria:

- CarPriority channel exists
- Collision presets configured correctly
- Collision matrix documented

Task 3.2: AFakeCarBody Actor Creation

Estimated Effort: 2 days
Priority: Critical
Dependencies: Task 3.1

- [ ] Create AFakeCarBody class

    - [ ] Inherit from AActor
    - [ ] Add UStaticMeshComponent (kinematic body)
    - [ ] Add UBoxComponent (overlap detection)
    - [ ] Add reference to owner car

- [ ] Implement BeginPlay

    - [ ] Copy mesh from owner car
    - [ ] Setup kinematic body collision
    - [ ] Lock all physics constraints
    - [ ] Hide mesh in game
    - [ ] Setup overlap detection box
    - [ ] Bind overlap events

- [ ] Implement AsyncPhysicsTick

    - [ ] Sync position with owner car
    - [ ] Sync rotation with owner car

Acceptance Criteria:

- Fake body spawns correctly
- Mesh is hidden
- Syncs with player car accurately
- No visible lag or desync

Task 3.3: Kinematic Body Physics Setup

Estimated Effort: 1 day
Priority: Critical
Dependencies: Task 3.2

- [ ] Configure kinematic mesh
  - [ ] SetSimulatePhysics(true)
  - [ ] SetEnableGravity(false)
  - [ ] Lock all translation axes
  - [ ] Lock all rotation axes
- [ ] Set collision properties
  - [ ] Object type: WorldDynamic
  - [ ] Collision enabled: QueryAndPhysics
  - [ ] Block WorldDynamic (AI cars)
  - [ ] Ignore everything else
- [ ] Test physics interaction
  - [ ] AI cars collide with fake body
  - [ ] AI cars are pushed away
  - [ ] Player car unaffected

Acceptance Criteria:

- Kinematic body doesn't move independently
- AI cars are pushed by fake body
- Player car trajectory unchanged

Task 3.4: Overlap Detection & Eight Cases

Estimated Effort: 1.5 days
Priority: High
Dependencies: Task 3.3

- [ ] Create ECollisionCase enum
  - [ ] FrontLeft, FrontRight
  - [ ] RearLeft, RearRight
  - [ ] LeftSide, RightSide
  - [ ] DirectFront, DirectBack
- [ ] Implement DetermineCollisionCase
  - [ ] Calculate impact direction
  - [ ] Calculate forward/right dot products
  - [ ] Classify into one of 8 cases (including direct front/back)
- [ ] Implement OnCarOverlap
  - [ ] Get impact direction

    - [ ] Determine collision case
    - [ ] Trigger visual effects

Acceptance Criteria:

- All 8 cases detected correctly
- Detection accuracy >95%
- No false positives

Task 3.5: Visual Shake Effects

Estimated Effort: 1 day
Priority: Medium
Dependencies: Task 3.4

- [ ] Create camera shake assets
    - [ ] Front corner shake (strong)
    - [ ] Rear corner shake (medium)
    - [ ] Side shake (light)
- [ ] Create FCollisionShakeSettings
    - [ ] References to shake assets
    - [ ] Intensity multiplier
- [ ] Implement ApplyVisualShake
    - [ ] Select shake based on collision case
    - [ ] Apply to player controller
    - [ ] Scale intensity appropriately

Acceptance Criteria:

- Shake effects feel impactful
- Different shakes for different cases
- Not disorienting

Task 3.6: Testing & Polish

Estimated Effort: 1 day
Priority: High
Dependencies: Tasks 3.1-3.5

- [ ] Unit tests
    - [ ] Collision channel validation
    - [ ] Fake body sync accuracy
    - [ ] Eight case detection
- [ ] Integration tests

    - [ ] Player pushes AI cars
    - [ ] AI cannot push player
    - [ ] Works during various states

- [ ] Manual testing

    - [ ] Feels powerful
    - [ ] AI resistance appropriate
    - [ ] Performance on mobile

Acceptance Criteria:

- All tests pass
- Player advantage clear
- 30 FPS Android, 60 FPS iOS

Phase 4: ME08 - Ramp & Airborne Completion (3-4 days)

Task 4.1: Ramp Zone System

Estimated Effort: 1 day
Priority: High
Dependencies: None

- [ ] Create ARampZone actor

    - [ ] Add UBoxComponent trigger
    - [ ] Add boost force property
    - [ ] Note: Boost angle follows vehicle's velocity direction (not a fixed property)
    - [ ] Note: Boost height determined by force and angle, not directly defined

- [ ] Implement OnRampOverlap

    - [ ] Cancel drift if active
    - [ ] Calculate boost direction (follows velocity direction)
    - [ ] Apply fixed force impulse
    - [ ] Trigger camera effects
    - [ ] Set airborne state

- [ ] Tune boost force and angle

    - [ ] Achieve 4-6m height
    - [ ] Adjust force angle higher than ramp surface angle to increase height
    - [ ] Test various car speeds

Acceptance Criteria:

- Ramp boost achieves 4-6m height
- Boost direction follows vehicle velocity
- Drift cancels on entry

Task 4.2: Air Control Refinement

Estimated Effort: 0.5 day
Priority: Medium
Dependencies: Task 4.1

- [ ] Create FAirControlSettings
  - [ ] Air steering multiplier: 0.5
  - [ ] Yaw strength only (no pitch/roll control)
- [ ] Implement ApplyAirControl
  - [ ] Get player steering input
  - [ ] Calculate yaw torque only
  - [ ] Apply 50% reduction
  - [ ] Player can only control yaw (left/right rotation) in air
- [ ] Test air steering
  - [ ] Yaw control feels responsive
  - [ ] Limited enough for challenge

Acceptance Criteria:

- Air yaw steering is 50% of ground
- Player can only control left/right rotation (no pitch/roll)
- Feels controllable but limited
- Positive playtest feedback

Task 4.3: Auto-Rotate System

Estimated Effort: 1 day
Priority: High
Dependencies: Task 4.2

- [ ] Create FAutoRotateSettings
  - [ ] Max roll angle: 45°
  - [ ] Max pitch angle: 90°
  - [ ] Rotation speed
  - [ ] Correction torque
- [ ] Implement ShouldAutoRotate
  - [ ] Check roll >45°
  - [ ] Check upside down
- [ ] Implement ApplyAutoRotate
  - [ ] Calculate target rotation (upright)
  - [ ] Smooth interpolation

    - [ ] Apply correction torque

Acceptance Criteria:

- Auto-rotate triggers at >45° roll
- Auto-rotate triggers when upside down
- Smooth correction (not jarring)

Task 4.4: Landing & Reset System

Estimated Effort: 1 day
Priority: High
Dependencies: Task 4.3

- [ ] Implement OnLanding
    - [ ] Check landing rotation
    - [ ] Apply landing effects if upright
    - [ ] Reset car if tilted
- [ ] Implement CheckUprightLanding
    - [ ] Max roll: 30°
    - [ ] Max pitch: 45°
- [ ] Implement ResetCarPosition
    - [ ] Find nearest road point
    - [ ] Reset position/rotation
    - [ ] Reset velocity
- [ ] Add camera bounce effect
    - [ ] Down 20cm on landing
    - [ ] Animate back over 0.3s

Acceptance Criteria:

- Upright landings feel satisfying
- Tilted landings reset correctly
- Camera bounce enhances feel

Task 4.5: Testing & Polish

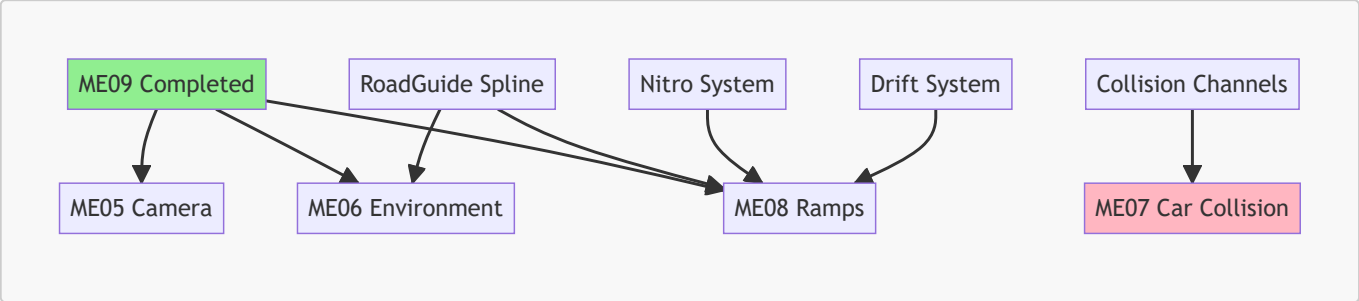
Estimated Effort: 0.5 day
Priority: High
Dependencies: Tasks 4.1-4.4

- [ ] Unit tests
- [ ] Integration tests (NOS, drift)
- [ ] Manual testing
- [ ] Performance validation

Acceptance Criteria:

- All tests pass
- Exciting gameplay
- Mobile performance targets met

Dependencies



Timeline & Estimates

Phase	Feature	Estimated Days	Priority
1	ME05 - Camera	3-5	High
2	ME06 - Environment	2-3	High
3	ME07 - Car Collision	5-7	Critical
4	ME08 - Ramps	3-4	High

Total: 13-19 days (2.5-4 weeks)

Buffer: +20% for unknowns = 16-23 days

Risks & Mitigation

Risk	Impact	Probability	Mitigation
ME07 collision channel conflicts	High	Medium	Thorough testing, document matrix, validate early
Mobile performance degradation	High	Medium	Profile early, distance-based updates, adaptive quality
Ramp force tuning iterations	Medium	High	Adjustable parameters, rapid iteration, designer tools
Camera interpolation unnatural	Medium	Medium	Expose curves, designer control, user testing
Kinematic body sync lag	Medium	Low	Async Physics Tick, validate timing, stress test
Auto-rotate feels intrusive	Low	Low	Tunable thresholds, optional disable, playtest

Resources Needed

Tools

- Unreal Engine 5.x
- Android/iOS test devices
- Profiling tools (Unreal Insights)

Assets

- Camera shake effects (ME07)
- VFX for ramps (ME08)
- Audio cues (optional)