VectorVelocity: Hover Vehicle Template

A complete, multiplayer-ready hover vehicle kit for Unreal Engine 5, built for Blueprint.

This template gives you a flying car that already drives, hovers, boosts, takes damage, shows a HUD, and even has enemies. You build your game by changing settings in panels and connecting nodes in Blueprint. You do not need to write any code.

This guide explains every part in plain language and shows you where to click. If you are new to Unreal, read it top to bottom. If you have used Unreal before, jump to the section you need.

Table of Contents

1. What You Get

VectorVelocity is a starting point for hover racing or combat games, like Wipeout or F-Zero. The hard parts (physics, online play, smoothing) are already done. Your job is the fun part: pick a car model, change numbers until it feels right, add sounds and effects, and build levels.

Here is what is in the box:

- **Hovering that feels real.** The car floats above the ground using invisible "legs" that push it up. It leans on hills, soaks up bumps, and lifts itself if the ground gets too close.
- **Gears.** Like a real car, it has Reverse, Neutral, and forward gears 1 to 5. Each gear has its own top speed and feel. You can add or remove gears in a simple list.

- **A boost (dash).** Tap a button for a burst of speed that smoothly ramps up, peaks, then fades, followed by a short cooldown.
- **Health and crash damage.** Hit a wall fast and you take damage. Hit it very fast and the car is destroyed. You can heal or repair from Blueprint.
- **A cinematic camera.** The view pulls back and widens as you speed up, leans into turns, and can shake at high speed.
- **Body lean.** The car tips back when you boost and leans into corners. This is only visual, so it never messes up the driving.
- **A HUD (the on-screen display).** Shows speed, gear, health, and boost. You design how it looks; the template fills in the numbers.
- **Effects (Niagara).** Slots for thruster flames, boost trails, brake sparks, speed lines, and one-time effects for crashes and landings. You drop in your effect; the template turns it on and off at the right time.
- **Enemies.** A chasing enemy hover car and a guard turret that aims at you. Both are ready to drop into a level.
- **One node for all stats.** A single Blueprint node gives you speed, health, gear, boost, and more, all at once.
- **Ready controls.** Keyboard, mouse, and gamepad already work, so you can press Play right away.

2. Words You Will See (Quick Glossary)

If a word below is new to you, come back here.

- **Blueprint (BP):** Unreal's visual scripting. You connect boxes (nodes) instead of typing code.
- **Component:** A reusable part you attach to an actor. This template's features (hover, gears, health, camera) are each a component.
- **Actor:** Anything you can place in a level. A vehicle, a turret, a light.
- **Pawn:** An actor a player or AI can control. Your car is a pawn.
- **Details panel:** The panel on the right of the editor where you change an object's settings.
- **Property / setting:** One value you can change, like "Max Speed."
- **Node:** A single action or value in a Blueprint graph.
- **Event:** A node that runs when something happens, like "when the car is destroyed." You add your own logic after it.
- **Event Dispatcher:** A signal a component sends out. Other Blueprints can listen and react. You "Bind" to it.
- **Alpha:** A number from 0 to 1 used as a percentage. 0 means empty or none, 1 means full. Example: health alpha 0.5 means half health.
- **cm/s:** Centimeters per second. Unreal measures distance in centimeters, so speeds are in cm/s. (Multiply by 0.036 to get km/h.)
- **Replication / replicated:** Sharing information between the server and players in multiplayer so everyone sees the same thing.
- **Server-authoritative:** The host machine (server) makes the real decisions. This stops cheating and keeps everyone in sync. The template handles this for you.
- **Collision channel:** A label Unreal uses to decide what blocks or detects what. The hover legs use one to find the ground.

- **Curve asset (Curve Float):** A small asset that draws a line on a graph. The boost can read one to shape how it ramps up and down.
- **Niagara:** Unreal's effects system for particles like fire, sparks, and smoke.
- **Niagara System:** One finished Niagara effect asset that you drop into a slot.
- **UMG:** Unreal's UI builder, where you design the HUD by dragging boxes and bars.
- **Enhanced Input:** Unreal's modern input system, which maps keys and buttons to actions.
- **Spring Arm:** An invisible pole that holds the camera behind the car and smooths its motion.
- **NavMesh:** A map of where AI can walk. Good news: the chasing enemy here does not need one.

3. Technical Details

Item	Value
Engine version	Unreal Engine 5.2
Main workflow	Blueprint (full nodes and events; C++ source is included but you do not have to touch it)
Ready-to-use classes	12 (vehicle, AI vehicle, AI brain, turret, 6 components, HUD, game mode)
Multiplayer	Yes. Built in and server-authoritative.
Input	Enhanced Input. Controls already set up.
Platforms	Set up for Windows (DX12). The design is platform-neutral and should build elsewhere, but test before you ship.
Plugins used	Enhanced Input, Niagara, AIModule, NavigationSystem, Chaos. All come with the engine. No paid plugins.
Demo content	A demo level, a drivable car Blueprint, a HUD, an enemy car, a turret, and Epic's Starter Content.

The classes you will actually use

A "class" here is just a ready-made building block. Most have a demo Blueprint version (in **bold**) that you can open and change.

Class	What you do with it
HoverVehiclePawn (demo: **Bp_Player**)	Your drivable car. Open it, pick a mesh, change settings.
HoverVehicleAIPawn (demo: **HoverAi**)	An enemy car. Drag it into a level and it chases the player.
HoverVehicleAIController	The enemy's "brain." Change how it detects and chases.
TurretAI (demo: **TurretAi**)	A guard turret. Drag it in, then tell it what to do when it fires.
HoverComponent	The hovering. Sets the legs, ride height, and stiffness.
VehicleMovementComponent	Driving: thrust, turning, boost, brake.
GearSystemComponent	The gear list and auto-shift options.
VehicleHealthComponent	Health and crash damage.
VehicleCameraComponent	The cinematic camera.

Class	What you do with it
VehicleFXComponent	Effect slots (you add this one when you want effects).
VectorVelocityHUDWidget (demo: **WBP_VehicleHUD**)	The HUD. You design the look.
VectorVelocityGameModeBase (demo: **Bp_VelocityGamemode**)	Tells the level which car to spawn.

Everything above can be opened and edited in Blueprint. Every setting is in a panel. Every action is a node. Every important moment is an event you can react to.

---

4. Before You Start

You need two things:

1. **Unreal Engine 5.2.** Install it from the Epic Games Launcher.
2. **A C++ compiler, installed once.** On Windows this is **Visual Studio 2022** with the **"Game development with C++"** option ticked during install. You do not write C++. The engine just needs the compiler to build the template's code one time.

Why does a Blueprint template need a compiler? The features are written in C++ for speed and smooth online play, then opened up to Blueprint. It is like a plugin: it builds once, and after that you stay in Blueprint forever. If the editor asks to build when you open the project, that is normal. Click Yes and wait a minute.

---

5. First Launch, Step by Step

1. **Open the project.** Double-click `VectorVelocity.uproject` . If a window says some modules are missing and asks to rebuild, click **Yes**. This is the one-time build. Wait for it to finish.
2. **The editor opens on the demo level.** The level is at `Content/VectorVelocity/Maps/Demo` . You can see it in the **Content Browser** at the bottom.
3. **Press Play.** Use the big **Play** button in the toolbar. You will spawn inside a working hover car, with the HUD already showing.
4. **Drive around.** Use the controls in the next section.
5. **Try multiplayer (optional).** Click the small arrow next to Play to open **Play settings**. Set **Number of Players** to 2 and **Net Mode** to **Play As Listen Server**, then press Play. Two windows open, each driving its own car, fully synced.

That is the whole loop. From here you change things and press Play again to test.

---

6. The Controls

These already work. You do not have to set them up.

What you want to do	Keyboard / Mouse	Gamepad
Speed up / slow down	W / S	Left Stick up / down
Turn left / right	A / D, or move the Mouse left/right	Left Stick left / right
Raise / lower hover height	E / Q	Right Trigger / Left Trigger
Boost (dash)	Left Shift	Right face button (B or Circle)
Brake	Spacebar	Bottom face button (A or Cross)
Shift up a gear	Mouse Wheel up	Right Shoulder (RB)
Shift down a gear	Mouse Wheel down	Left Shoulder (LB)

About gears: You usually do not need the gear buttons. By default the car shifts for you. Hold S while moving forward and it drops into reverse on its own. Hold W while reversing and it climbs back into forward gears. The manual gear buttons are there if you want them.

To change the controls: Open `Bp_Player` , select it, and in the Details panel find the **Vehicle - Input** section. Drop in your own Input Mapping Context and Input Action assets. Anything you set replaces the default; anything you leave empty keeps the default.

---

7. How a Vehicle Is Built

Open `Bp_Player` (double-click it in the Content Browser at `Content/VectorVelocity/Player/`). On the left you will see the **Components** panel. It looks like this:

```
Bp_Player  (the car)
  ChassisCollision  (a box)        the real physics body, everything hangs off this
    MeshPivot                      holds the visual lean; never affects driving
      VehicleMesh                  put YOUR car model here
    Hover         (HoverComponent)         the hovering
    Movement      (VehicleMovementComponent) driving and boost
    HealthComp    (VehicleHealthComponent)   health and damage
    GearSystem    (GearSystemComponent)       the gears
    CameraRig     (VehicleCameraComponent)    the camera
  SpringArm                         the camera pole
  Camera                            the actual camera
```

What each piece does, in plain terms:

- **ChassisCollision** is the solid, invisible box that actually moves and bumps into things. Everything else rides on it.
- **VehicleMesh** is the car model you see. You will replace this with your own model. It sits under **MeshPivot** so it can lean without changing how the car drives.
- The **components** (Hover, Movement, HealthComp, GearSystem, CameraRig) are the features. Each has its own settings in the Details panel.
- **SpringArm** and **Camera** make the third-person view.

The effects part, **VehicleFXComponent**, is not on the car by default. You add it yourself when you want flames and trails (see the Effects section).

A few ideas that make this template easy

- **Each feature is separate.** Do not want gears? Delete the GearSystem component. Want different hovering? Replace the Hover component. You keep what you need.
- **The look and the physics are kept apart.** Leaning, banking, and camera moves only touch the visuals. You can make them dramatic without breaking the driving.
- **There is an event for almost everything.** When the car lands, gets destroyed, changes gear, or hits something, the template fires an event. You add your sound or effect right there in the Event Graph.

---

8. Every Setting Explained

To change these: open the vehicle Blueprint, click the component in the Components panel, and edit the Details panel on the right. The numbers below are the defaults that ship with the template.

Many of the most-used settings are also copied to the top of the car itself, under **Vehicle - Quick Tuning**, **Vehicle - Tilt**, and **Vehicle - Layout**, so you can change them without digging into each component.

Quick Tuning (on the car, easiest place to start)

Setting	Default	What it does
Chassis Box Extent	150, 90, 30	Half the size of the invisible physics box. Make it match your car model.
Chassis Mass	1500 (kg)	How heavy the car is. Heavier cars need more push to move and hover.
Hover Lift Strength	2400	How hard the legs push the car up.
Hover Height	120 (cm)	How high the car floats when resting.
Vertical Thrust Strength	800	How fast Q/E raise or lower the car.
Turning Speed	130 (degrees/sec)	How fast it turns at full input.

Tilt (the visual lean)

These only change how the car looks while driving. Bigger numbers look more arcade-like.

- **Pitch Tilt 14:** how much it tips back when boosting or noses down when braking.
- **Roll Tilt 38:** how much it leans into turns. Raise this for a strong arcade lean.
- **Yaw Rate For Full Bank 70:** how sharp a turn must be to reach full lean.
- **Velocity Smoothing 8** and **Tilt Interp Speed 9:** how smooth or snappy the lean feels.
- **Vertical Tilt 12:** how much it tips when climbing or dropping in height.

Layout

By default you move and rotate **VehicleMesh**, **MeshPivot**, and **CameraRig** by hand using the move tool in the Blueprint. That is the normal way.

There is an option called **Apply Layout Overrides**. Leave it **off** unless you want the car to set those positions from number fields instead (useful for advanced, automated setups). If you turn it on, it will overwrite the positions you set by hand.

Hover Component (the hovering)

Setting	Default	What it does
Hover Height	120 (cm)	Target float height.
Hover Force	2400	How hard each leg pushes up.
Hover Damping	220	Controls bounce. Higher is stiffer and steadier; lower is floatier and softer over bumps.
Max Trace Distance	600 (cm)	How far down the legs look for the ground, at minimum.
Extra Trace Buffer	400 (cm)	Extra looking distance so the legs still find the ground when the car rises.
Hover Points	4 corners	The list of legs. 4 acts like a car, 2 like a bike, 1 like a floaty drone. You can add or move them.
Ground Trace Channel	Visibility	Which collision type counts as "ground." Match this to your floor.
Align To Ground Normal	on	Tilt the car to follow slopes and ramps.
Align To Ground Speed	7	How quickly it follows slopes.
Auto Correct On Low Clearance	on	Automatically lift the car if the ground gets too close.
Min Ground Clearance	40 (cm)	The closest the car may get before it auto-lifts.
Auto Correct Speed	250 (cm/s)	How fast it auto-lifts.
Draw Debug	off	Turn on to see the hover legs in the viewport while testing.

What is a "hover point"? It is one invisible leg. The car shoots a line straight down from each leg, measures how far the ground is, and pushes up like a spring. More legs spread out means a steadier car.

Vehicle Movement Component (driving and boost)

Setting	Default	What it does
Acceleration	2400	Base push when speeding up (before the gear changes it).
Max Speed	4500 (cm/s)	Top speed before a gear sets its own limit.
Turn Speed	130 (degrees/sec)	Turn rate at full input.
Vertical Force	800	How fast Q/E change hover height.
Drag Coefficient	0.025	Air resistance. Lower means more gliding and coasting.
Reverse Scale	0.5	Reverse power compared to forward. 0.5 is half power.
Dash Multiplier	2.5	How strong the boost is at its peak (2.5 times).
Dash Duration	2.0 (sec)	How long one boost lasts, start to finish.
Dash Cooldown	2.0 (sec)	The wait before you can boost again.
Dash Curve	empty	Optional Curve asset to shape the boost. Empty uses a smooth built-in bell shape.

Setting	Default	What it does
Brake Force	2500	How hard the brake slows you down.

How the boost works: It does not just snap on. It ramps up to its peak in the middle, then fades back down over the Dash Duration. If you want a custom shape (instant punch, slow build, etc.), make a Curve Float asset and drop it in Dash Curve.

Gear System Component (the gears)

The gears live in a list called **Gears**. Each row is one gear with these fields:

- **Gear Name:** a label for you (like "First").
- **Display Label:** the short text shown on the HUD (R, N, 1, 2, and so on).
- **Max Speed:** the top speed in this gear.
- **Acceleration Multiplier:** how strong the push is in this gear.
- **Turn Multiplier:** how sharp the turning is in this gear (lower gears usually turn tighter).

Setting	Default	What it does
Gears	R, N, 1 to 5	The list of gears. Add or remove rows freely.
Neutral Gear Index	1	Which row is Neutral (coasting).
Reverse Gear Index	0	Which row is Reverse. Set to -1 to remove reverse.
Starting Gear	1	The gear the car spawns in.
Auto Return To Neutral	on	Drift back to Neutral when you let go of the gas.
Auto Return Delay / Interval	1.5 / 0.6 (sec)	How long before it drifts, and how fast.
Auto Shift On Input Direction	on	Hold S to go into reverse; hold W to go forward.
Auto Shift Interval	0.5 (sec)	How fast it auto-shifts.

(An "index" is just the row number in the list, starting at 0. So Reverse Gear Index 0 means the first row.)

Vehicle Health Component (health and crash damage)

Damage is figured out automatically from how fast you hit something. You do not wire anything up.

Setting	Default	What it does
Max Health	100	The size of the health bar.
Damage Multiplier	1.0	Scales all damage. 2.0 means double damage.
Min Damage Speed	600 (cm/s)	Below this speed, bumps do no damage.
Max Damage Speed	2500 (cm/s)	At this speed, hits do full damage.
Instant Destroy Speed	4000 (cm/s)	Hit something this fast and the car is destroyed at once.
Max Impact Damage	80	The damage dealt at Max Damage Speed.

Vehicle Camera Component (the camera)

Group	Settings (defaults)
Distance	Min Arm 520, Max Arm 900, Hard Min 250, Hard Max 1400, Interp Speed 3.5

Group	Settings (defaults)
Field of View	Min 88, Max 118, Interp Speed 4.5
Lag (smoothing)	Camera Lag 10, Rotation Lag 6
Cinematic	Look-Ahead -220, Speed Pitch -10, Bank Follow 0.55, Interp Speed 4.5
Shake	Shake Start Alpha 0.6 (the point where the shake event starts firing)

In plain terms: the camera sits closer and at a normal view when slow, then pulls back, widens, dips down, and follows the lean as you speed up. "Field of View" is how wide the camera sees; wider feels faster. The "Hard Min/Max" numbers stop the camera from ever clipping into the car or flying too far away.

Vehicle FX Component (effects)

This holds slots for your effects. Empty slots simply do nothing, so you only fill in what you want.

- **Always-on slots (you assign a Niagara System; the template turns each on and off):** Hover Thrusters System, Dash Trail System, Brake Sparks System, Speed Trails System.
- **Position slots:** Dash Trail Offset and Brake Sparks Offset move the rear emitters.
- **One-time slots:** Impact (on a crash), Landing (when you touch down), Destroyed (when health hits 0).
- **Tuning:** Speed Trail Threshold 0.7 (how fast before speed lines appear), Min Impact FX Speed 400, Min Landing FX Speed 250.

---

9. Blueprint Nodes and Events

These are the actions and signals you use in Blueprint. The names are exactly what you will see when you search in a graph. To use one, drag a wire from the car or the component and type the name.

A **node** does or reads something. A **Pure** node just reads a value (green, no run order). An **Event** runs when something happens, and you add your logic after it. An **Event Dispatcher** is a signal you "Bind" to so you can react.

The one node to remember

Get Vehicle Stats (Pure, on the car). This gives you everything at once in a single package: Speed, Speed Alpha, Max Speed, Health, Max Health, Health Alpha, Is Destroyed, Current Gear, Gear Label, Is In Neutral, Is In Reverse, Is Dashing, Dash Available, Dash Charge Alpha, Dash Curve Alpha, Dash Speed Multiplier, Is Braking, Is Grounded, Current Hover Height, Is Auto Correcting, and the Vehicle State. It is cheap to use every frame, which makes it perfect for the HUD.

On the car

- **Get Vehicle State** (Pure): tells you if the car is Idle, Driving, Boosting, Braking, Airborne, or Destroyed.
- **Set AI Input** and **Get Current Input**: feed or read the drive input. Used to control a car from your own scripts.
- **Event On Vehicle State Changed** (gives New State): great for engine pitch, lights, or UI changes.
- **Event On Input Gathered** (gives Input): fires for the local player when input is read.

Movement

- **Trigger Dash** (returns true if it started), **Cancel Dash**, **Set Dash Cooldown Remaining** (handy for a "refill boost" pickup).
- **Get Current Speed**, **Get Speed Alpha**, **Is Dashing**, **Is Dash Available**, **Get Dash Charge Alpha**, **Get Dash Curve Alpha**, **Get Dash Speed Multiplier**, **Is Braking**, **Get Effective Max Speed**, **Get Effective Acceleration** (all Pure reads).
- **Event Apply Additional Forces** (gives Delta Time): add your own push here, like a boost pad or wind.

Hover

- **Is Grounded**, **Get Compression Alpha**, **Get Average Ground Normal**, **Get Current Target Height**, **Is Auto Correcting** (Pure reads).
- **Adjust Height** (add or subtract), **Set Target Height**, **Reset Height**.
- **Event On Grounded State Changed** (gives Now Grounded): fire landing dust or a thump sound.

Gears

- **Get Current Gear**, **Get Gear Count**, **Get Current Gear Settings**, **Is In Neutral**, **Is In Reverse** (Pure reads).
- **Increase Gear**, **Decrease Gear**, **Set Gear**.
- **On Gear Changed** (Event Dispatcher, gives New Gear): bind it for a shift sound or HUD flash.

Health

- **Get Health**, **Get Max Health**, **Get Health Alpha**, **Is Destroyed** (Pure reads).
- **Apply Damage**, **Heal**, **Set Health**, **Set Max Health** (with a "keep the same percentage" option), **Refill Health**.
- **On Health Changed** (Dispatcher: New Health, Max Health), **On Destroyed** (Dispatcher), **Event On Destroyed BP**: spawn a wreck, start a respawn timer, update score.

Camera and Effects

- **Event On High Speed Shake** (gives Speed Alpha): add your camera shake here.
- **Play Impact FX** (Location, Normal), **Play Landing FX**, **Play Destroyed FX**: fire one-time effects yourself if you want.
- **Event On FX Tick** (gives Speed Alpha, Grounded, Dashing, Braking): drive any custom effect from live state.

---

10. The Enemies (AI)

The chasing enemy car: HoverAi

Find it in `Content/VectorVelocity/Ai/` . Drag it into your level and press Play. It wanders near where you placed it. The moment the player gets close, it locks on and chases, circling and strafing to keep a good distance. Once it has seen you, it never goes back to calmly wandering.

The clever part: the enemy drives using the same input path a human uses. So it automatically gets all the smooth hovering, gear changes, and boosting that you do. It does not need a NavMesh, so it works fine in open arenas.

To change how it behaves, open the AI and look in the Details panel. The main groups:

- **Detection:** Detection Range 4000 (how close you must be), and Target Tag (explained below).
- **Combat:** Ideal Distance 1500 (the gap it likes to keep), Min Distance 800 (too close, it backs off), Dash Trigger Distance 3000 (far away, it boosts to catch up).
- **Roam:** how far it wanders and how long it pauses.
- **Reposition:** how often and how sharply it circles you.
- **Steering, Altitude, Gears:** how sharp it turns, whether it matches your height, and which gears it uses.
- **Event On AI State Changed:** add alert sounds or UI when it spots you.

To make different enemy types (a fast scout, a heavy bruiser), right-click `HoverAi` and choose **Create Child Blueprint Class**, then give each child its own model and settings.

The guard turret: TurretAi

Find it in `Content/VectorVelocity/Ai/` . Drag it into a level. It slowly sweeps back and forth on patrol. When the player enters its range, it aims at you and tracks you, even straight up if you fly overhead. In multiplayer it looks smooth on every screen.

Important: the turret does not shoot on its own. This is on purpose, so you decide what "firing" means. Open `TurretAi` , go to the Event Graph, and find the **On Fire** event. From there, spawn your projectile or do a line trace toward the target, play a sound, and add a muzzle flash. Set **Fire Interval** for how often it fires (0 turns off auto-fire).

Tip: telling the AI who to target

Both enemies have a **Target Tag** setting. If you leave it empty, they target any pawn except themselves. If you set it (for example, "Player"), they only target actors that carry that same **Actor Tag**. To tag your car, open `Bp_Player` , select the top of the Components list, and in Details add "Player" under **Actor - Tags**. Then set the AI's Target Tag to "Player."

---

11. Setting Up the HUD

The HUD is the on-screen display for speed, gear, health, and boost. You design how it looks; the template fills in the numbers.

1. **Use the demo or make your own.** The demo HUD is `WBP_VehicleHUD` in `Content/VectorVelocity/UI/` . To make a fresh one, right-click in the Content Browser, choose **User Interface, Widget Blueprint**, and when asked for the parent pick **VectorVelocityHUDWidget**.
2. **Add the widgets and name them exactly.** Open your HUD in the UMG Designer. Drag in any of these and give them these exact names (any you skip are simply ignored, no errors):

- `SpeedText` and `GearText` (Text)
- `HealthBar` , `DashBar` , `SpeedBar` (Progress Bars; they expect a value from 0 to 1)

1. **Make it look how you want.** Position, color, font, and animate freely.
2. **Tell the car to use it.** Open `Bp_Player` , select it, and in Details find **Vehicle - HUD - HUD Widget Class**. Pick your HUD. It appears automatically for the local player. (Leave it empty to hide the HUD.)

Want full control? In your HUD's Event Graph, add the **On HUD Update** event. It runs every frame and hands you Speed, Speed Alpha, Current Gear, Gear Name, Health, Max Health, Health Alpha, Dash

Charge Alpha, and whether you are dashing. Use these to format text, change colors, or play animations. The Details panel also has **Speed Format** (the text style, like "{0} KM/H"), **Speed Display Multiplier** (0.036 turns cm/s into km/h), and **Gear Format**.

12. Setting Up the Effects (Niagara)

Niagara is the particle system (flames, sparks, trails). The template knows when to play effects; you provide what the effect looks like. You assign a finished Niagara System to each slot, and the template creates and controls the emitter for you.

1. **Add the effects component.** Open your car Blueprint, click **Add Component**, and choose **VehicleFXComponent**. Drag it under ChassisCollision in the Components list.
2. **Fill the always-on slots.** Select the VehicleFXComponent. In its Details panel, find the **FX - Persistent** section and drop a Niagara System into each slot you want: **Hover Thrusters System**, **Dash Trail System**, **Brake Sparks System**, and **Speed Trails System**. The template builds and toggles these for you as you drive.
3. **Set where the trails sit.** In the same section, **Dash Trail Offset** and **Brake Sparks Offset** move those emitters back behind the car. Change them to fit your model.
4. **Fill the one-time slots.** Set the **Impact**, **Landing**, and **Destroyed** slots. These fire once on a crash, on a landing, and when the car is destroyed.
5. **Adjust the triggers (optional). Speed Trail Threshold** sets how fast you must go before speed lines appear. **Min Impact FX Speed** and **Min Landing FX Speed** set how hard a hit or landing must be to show an effect.

That is all. Turning effects on and off, and syncing them in multiplayer, is automatic. Any slot you leave empty stays silent. You build the actual particle look in the Niagara editor (there are many free Niagara packs to start from).

13. Multiplayer

The template is built for online play, and you do not have to manage any of it.

- **The server runs the driving; players see the result.** You build your car once and it works the same alone or online.
- **The HUD and effects get the right info on every screen.** Health, gear, boost, braking, hover height, and grounded state are all shared, so effects and UI look correct for everyone, not just the local player.
- **AI and turrets run on the server**, and their motion is shared with smoothing built in.

To test: click the arrow next to Play, set **Number of Players** to 2 or more, and **Net Mode** to **Play As Listen Server**. Each window controls its own synced car.

If you add your own multiplayer logic: keep big decisions (damage, score, respawns) on the server. The built-in nodes already do this. When you add your own, use a **Switch Has Authority** node so the important part runs on the server.

14. Step-by-Step Guides

Put your own car model in

1. Open `Bp_Player` .
2. Click **VehicleMesh** in the Components panel.
3. In Details, set the **Static Mesh** to your model.
4. Use the move and rotate tools to line it up (it sits under MeshPivot).
5. Click the car (top of the list) and adjust **Chassis Box Extent** and **Chassis Mass** so the invisible box matches your model.
6. Press Play to check it.

Make it feel faster, heavier, or floatier

- **Faster:** raise **Max Speed** (and each gear's Max Speed). Raise the camera's **Max FOV** for a stronger sense of speed.
- **Heavier:** raise **Chassis Mass**, then raise **Hover Force** to hold it up, raise **Hover Damping**, and lower **Acceleration**.
- **Floatier:** lower **Hover Damping**, lower **Drag Coefficient**, and raise **Extra Trace Buffer**.

Add or change a gear

1. Open `Bp_Player` , select **GearSystem**.
2. In Details, expand **Gears**.
3. Click **+** to add a row, or edit an existing one.
4. Set the Display Label, Max Speed, Acceleration Multiplier, and Turn Multiplier.
5. Make sure **Neutral Gear Index** and **Reverse Gear Index** still point at the right rows.

Shape the boost with a curve

1. In the Content Browser, right-click, choose **Miscellaneous, Curve**, then **Curve Float**.
2. Open it. The bottom (X) is progress from 0 to 1; the side (Y) is boost strength from 0 to 1. Draw the shape you want.
3. Open `Bp_Player` , select **Movement**, and set **Dash Curve** to your curve.

Make the turret actually shoot

1. Open `TurretAi` , go to the **Event Graph**.
2. Find the **On Fire** event (it gives you the target).
3. From there, **Spawn Actor** for your projectile aimed at the target, or do a line trace and apply damage.
4. Add a sound and a muzzle flash.
5. Set **Fire Interval** for how fast it fires.

React when the car is destroyed

1. Open `Bp_Player` , go to the **Event Graph**.
2. Add the **Event On Destroyed BP** node (or, on the HealthComp, **Bind** to **On Destroyed**).
3. After it, spawn a wreck, hide the mesh, start a respawn timer, and update the score. This is normal node wiring.

Add a boost pad (custom push)

1. In `Bp_Player` , add the **Event Apply Additional Forces** node.
2. When the car overlaps your boost-pad volume, use **Add Force** on ChassisCollision.
3. Because this is added on top, the hover and steering keep working normally.

Drive a car from Blueprint (for a cutscene or remote control)

1. Make a **Vehicle Input State** variable (it has Thrust, Turn, Vertical, Dash Held, Brake Held).
2. Each frame, set the values you want.
3. Call **Set AI Input** on the car. It drives exactly as if a player or the AI were controlling it.

---

15. Problems and Fixes

The project asks to rebuild when I open it. That is the one-time C++ build. Click **Yes**. If it fails, install Visual Studio 2022 with the "Game development with C++" option, then reopen. After this you stay in Blueprint.

The car sinks into the floor or floats away. Check **Hover Height**, and check **Hover Force** against **Chassis Mass** (a heavy car needs more force). Make sure **Ground Trace Channel** matches your floor's collision. Turn on **Hover, Draw Debug** to see the legs while testing.

The camera pokes into the car or drifts too far. Lower or raise **Hard Min Distance** and **Hard Max Distance** on the camera component.

No HUD shows up. Set the car's **HUD Widget Class** to your HUD. The HUD only appears for the player you are controlling.

My effects do not play. Empty effect slots stay silent on purpose. Drop your Niagara Systems into the slots, and make sure you actually added a **VehicleFXComponent** to the car.

The enemy ignores my player. If you set a **Target Tag** on the enemy or turret, your car must have that same **Actor Tag**. Either match them, or clear the Target Tag so it targets any pawn.

My positions reset when I move parts in the Blueprint.

Apply Layout Overrides is turned on. Turn it off so you can move parts by hand.

---

16. Included Content and Licensing

The systems, the components, the AI, the HUD, and the demo Blueprints (`Bp_Player` , `Bp_VelocityGamemode` , `WBP_VehicleHUD` , `HoverAi` , `TurretAi`) are part of this template and are yours to use and change in your projects.

The project also includes **Epic's Starter Content** and some demo art, only to show the systems working. Before you sell a game:

- Replace the placeholder art (the demo car model, the level dressing) with your own, and make sure you are allowed to use anything you keep.

- If you do not use the Starter Content, you can remove it to shrink your game: delete the `Content/StarterContent` folder, then remove the `[StartupActions]` block from `Config/DefaultGame.ini` .

Treat the demo level art as a reference, not as final game art, unless you have checked the license.

17. Support

If you get stuck or find a bug, contact the template author and include:

- Your Unreal Engine version.
- What you expected, and what actually happened.
- The Output Log if it is an error (open it from the menu: **Window, Output Log**).

Thanks for using VectorVelocity. Have fun building.