1. Lateral Position

Lateral Position.png

1.1 Main goal

- Loop through lasers positioned on the left and right side of the car, get the road material that these lasers are hitting to calculate the vehicle lateral position.

1.2 Variables

- LaserObject
    - Explanation: A reference to all BP_laser objects, i.e., an array of lasers
    - Type: BP Laser Array
- LaserTemp
    - Explanation: A temporary variable is used to get the name of the laser being accessed
    - Type: BP Laser
- RoadType
    - Explanation: An array of strings stores the material that each laser is hitting
    - Type: String Array
    - Value: Each value of the array could either be empty, indicating no road material hit, or asphalt, indicating that a laser hits road material

1.3 Functions

- GET
    - Explanation: Get an array and an index, returns a temporary copy of the item in the array at that index
- LastHit
    - Explanation: Takes BP Laser as input, reads the last hit of that laser
- Break Hit Result
    - Explanation: Given a last hit by a laser (output of the LastHit function), extract different data from the hit result. The Lateral Positin function uses the “Hit Component” data extracted by this function
- Get Material from Collision Face Index
    - Explanation: Takes the “Hit Component” returned by the Break Hit Result function, retrieves the material applied to a particular collision mesh (the road type).
- Get Object Name

    - Explanation: Takes an object as input, returns its name
- Get Display Name
    - Explanation: Takes an object as input and returns its display name
- Join String Array
    - Explanation: Takes an array, returns a string containing each value of the array, separated by a specific separator (we are using ‘,’)
- Make Array
    - Explanation: Makes an array given inputs
- Print String
    - Explanation: Prints a string to the screen; we use this for debugging purposes
- Set Array Elem:
    - Explanation: Takes in an index and a value, adds it to an array

1.4 Flows/Logic

- The function loops through an array of laser objects, getting and recording the hit component of each laser. Then the function prints the name of the laser and its hit result to the screen, as well as makes an array of the hit results to be added to the .csv file

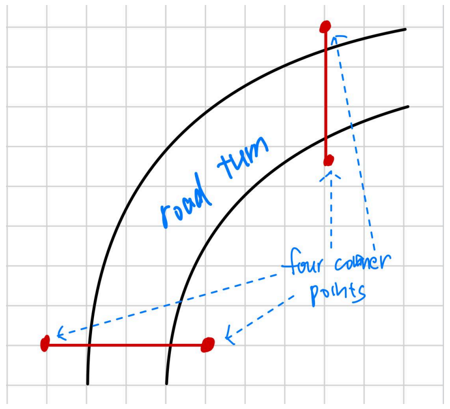
2. Curve Percentage

Curve Percentage

2.1 Main Goal

- Calculates and outputs the curve percentage, which is a value that indicates the progress along a curve or a turn.

2.2 Variables

- Cube Center Array
    - Explanation: stores the location of the four corner points at each turn. The corner points are located at the entry and exit of each turn.
        - 

    - Type: Array of vectors
- Vector Array
    - Explanation: The unit vectors of the cube center vectors
    - Type: Array of vectors
- Curve Data
    - Explanation: The curve percentage
    - Type: Float

2.3 Functions

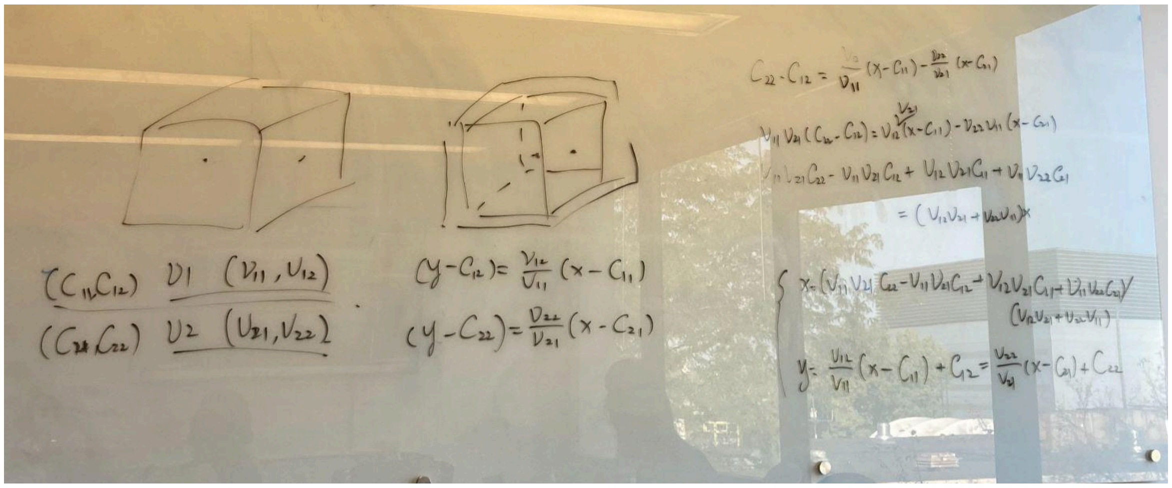
- GetActorLocation
    - Explanation: returns the location of an actor
- GET
    - Explanation: Given an array and an index, returns a copy of the item in the array at that index (value not reference, so changes made on the copy won't be reflected back to the original array)
- GetActorsWithTags
    - Explanation: Find all actors with the specified tag

- GetAttachedActors
  - Explanation: Takes in an actor and finds all actors directly attached to a component in said actor.
- GetUnitDirection
  - Explanation: Find the unit direction vector from one position to another
- BreakVector:
  - Explanation: Breaks a vector apart into X, Y, X
- MakeVector:
  - Explanation: Make a vector given inputs
- ACOSd
  - Explanation: Returns the inverse cos of input
- SetTimerByEvent
  - Explanation: Set a timer to execute delegate

2.4 Flows/Logic

- The functionality of calculating the curve percentage is realized by getting the four corner points (stored in the Cube Center Array). There are a pair of invisible walls at each turn (see drawing under the Cube Center Array variable); each pair is assigned a tag so that when one of the walls is hit (The vehicle enters a turn), we can get its tag and use the GetActorsWithTags function to get the other wall in the pair. From there, we can set the Cube Center Array and the Vector Array, which stores the unit vector of the Cube Center Array vectors. After we have set the Vector array and Cube Center Array, we compute the curve percentage by passing in the vector and the location of the vehicle into a series of complex math (math attached if you are interested in the details). After we compute the curve percentage, we store it in the Curve Data float variable and print it to the screen. Once the turn is complete (Curve Data is greater or equal to 1.0), clear the Vector array and Cube Center Array, and reset curve data to 0.

- Math for computing curve percentage:



$$(C_{11}, C_{12}) \quad V1 \quad (V_{11}, V_{12})$$
$$(C_{21}, C_{22}) \quad V2 \quad (V_{21}, V_{22})$$

$$(y - C_{12}) = \frac{V_{12}}{V_{11}}(x - C_{11})$$
$$(y - C_{22}) = \frac{V_{22}}{V_{21}}(x - C_{21})$$

$$C_{22} - C_{12} = \frac{V_{12}}{V_{11}}(x - C_{11}) - \frac{V_{22}}{V_{21}}(x - C_{21})$$
$$V_{11}V_{21}(C_{22} - C_{12}) = V_{12}V_{21}(x - C_{11}) - V_{22}V_{11}(x - C_{21})$$
$$V_{11}V_{21}C_{22} - V_{11}V_{21}C_{12} + V_{12}V_{21}C_{11} + V_{11}V_{22}C_{21}$$
$$= (V_{12}V_{21} + V_{22}V_{11})x$$

$$\begin{cases} x = (V_{11}V_{21}C_{22} - V_{11}V_{21}C_{12} + V_{12}V_{21}C_{11} + V_{11}V_{22}C_{21}) / (V_{12}V_{21} + V_{22}V_{11}) \\ y = \frac{V_{12}}{V_{11}}(x - C_{11}) + C_{12} = \frac{V_{22}}{V_{21}}(x - C_{21}) + C_{22} \end{cases}$$

3. Max Steer Angle

Max Steer Angle

3.1 Main Goal

- Control the maximum steering angle based on vehicle velocity; the faster the vehicle moves, the smaller the maximum steering angle is

3.2 Variables

- Simple Wheeled Vehicle Movement
  - Explanation: Get the movement of vehicle
  - Type: Object Reference

3.3 Functions

- Get Forward Speed
  - Explanation: Determine how fast the vehicle is moving forward from the Simple Wheeled Vehicle Movement reference
- Clamp (float)
  - Explanation: Take floats as input and return that float clamped between a Min and a Max inclusive: if the float is within A and B, return the float; if the float is less than the Min, return Min; and if the float is greater than the Max, return Max

3.4 Flows/Logic

- Get the forward speed of the vehicle, multiply it by 0,036 to convert from unreal units/second to kilometers/hour, and take the absolute value of the speed to account for reversing. Then multiply the speed by 0.6 and subtract it from 50, so the slower the speed, the greater the max steering angle is. Lastly, clamp the value to between 5.0 to 40.0 to ensure the maximum steering angle is within this range

4. Get RoadType Laser Option

Get RoadType LaserOption

4.1 Main Goal

- Determine whether the driver is on or off road. Apply vibration to the logitech steering wheel if the driver is off road since the driving surface is rough. Otherwise stop the vibration since the driving surface is smooth. Whether the driver is on or off road is determined by 4 virtual lasers attached to each truck's wheel. The driver is off road if at least 1 truck's wheel is off road.

Note: *This function was implemented by the 2023-2024 Data team. They hard coded the four lasers by accessing the laser array at indices 4, 5, 6, 7. The 2024-2025 Data team changed the laser configuration to prioritize the lane change experiment, which focused on determining the lateral lane position. Therefore, this function may no longer work as of Apr 12, 2025. However, this function **does not** cause the simulator to crash.*

Update 5/19/2025: This function works if the map loaded is Yehorivka. How it works is that the material of the road's name is hardcoded for each condition of the vibration (e.g. MI_Mainroad02 is a road material SPECIFIC for Yehorivka). If you want it to work in a different world, you need to find the name of the material type that the lasers pointing downwards will hit, and put the name in the Switch on String condition (located in GetTiredPeriod function of Military_Truck).

It is worth noting that the lasers are initiated in a weird order. This is a known bug. Unless the truck is compiled every time the player starts a new drive, the lasers might be spawned in a random order, potentially due to dynamic allocation. So, the 2023-2024 Data team's approach of hard-coding, which retrieves the lasers at fixed indices, might be buggy.

4.2 Variables

- **LaserObject**: Laser array, followed by the get method, which specifies the index to retrieve that specific laser. For example, LaserObject → Get(4) access the fifth laser because C++ indices start from 0.
- **Period**: How frequently should the vibration be, depending on how many wheels are off-road. Specifies the period of the periodic force effect. The value is the duration for one full cycle of the periodic function, measured in milliseconds. A good range of values for the period is 20 ms (sand) to 120 ms (wooden bridge or cobblestones). For a surface effect, the period should not be any bigger than 150 ms.
- **Rough Road**: Based on the material that the lasers are hitting, determine how rough the road is. This is mainly used to play off-road sound effects. More off road wheels lead to a rougher road.
- **Multiplier**: A scaler. Determined based on the material that the lasers are hitting. The rougher the road, the higher the multiplier. This scaler is used to scale **Sufface Eff Wheel.**
- **Sufface Eff Wheel**: Specifies the magnitude of the surface effect. Valid ranges for magnitudePercentage are 0 to 100. Values higher than 100 are silently clamped. In short, how aggressive (strong) should the vibration be.

4.3 Functions

- **GetTirePeriod**: Input is a single laser, which represents a wheel. Calculate the **period**, **rough road**, and **multiplier**, as specified as above.
- **CalcWheelFeedback**: Input is the **multiplier**, and the speed the truck is moving. Output is the **Sufface Eff Wheel.** In short, a high multiplier and a higher speed lead to a higher **Sufface Eff Wheel**
- **Play Wheel Surface Effect**: Takes in the **Sufface Eff Wheel** as Magnitude Percentage and **Period** as Period. Apply the vibration force to the Logitech steering wheel. The magnitude percentage and period determines how strong and frequent the vibration is, respectively.
- **Stop Wheel Surface Effect**: Stop the vibration.

4.4 Flows/Logic

- Go through each of the four lasers (each wheel) to calculate the cumulative **Suffface Eff Wheel** and **Period.** Then apply the vibration to the steering wheel to alert the player that they are driving on a rough surface; thus off-road. Otherwise, stop the vibration.