1. Brake Sound (written by Ian Yew)

Brake Sound

1.1 Variables

- Throttle
    - Explanation: is the representation of how much the pedal is pressed. If multiple inputs are being pressed, their moveForward values are added (if W and S are pressed, MoveForward = 0)
        - The example above is true if S is mapped to brake and not handbrake, handbrake trumps accelerator.
    - Type: Float
- Brake sound played
    - Explanation: Indication of whether brake sound has been played; if ticked (= false), it has not been played, otherwise, it has been played
    - Type: bool
- X
    - Type: float

1.2 Branches

- Branch 1:
    - If the throttle is greater than or equal to 0.0, set brake sound played to false. Otherwise, continue to branch 2.
- Branch 2:
    - If the absolute value of x is greater than or equal to 10.0, continue to branch 3. Otherwise, the program is terminated.
- Branch 3:
    - If a brake sound is played, the program is terminated. Otherwise, the Play Sound 2D function is called.

1.3 Functions

- Play Sound 2D
    - Arguments: sound, volume multiplier = 1.0, pitch multiplier = 1.0, start time = 0.0, concurrency settings, owning actor, Is UISound = true
    - Explanation: sets brake sound played to true

1.4. Brake Sound-SUV (written by Andreas Andrade)

BrakeSoundSUV.png

1.4.1 Variables

- Throttle
    - Explanation: is the representation of how much the pedal is pressed. If multiple inputs are being pressed, their moveForward values are added (if W and S are pressed, MoveForward = 0)
        - The example above is true if S is mapped to brake and not handbrake, handbrake trumps accelerator.
    - Type: Float
- Brake sound played
    - Explanation: Indication of whether brake sound has been played; if ticked (= false), it has not been played, otherwise, it has been played
    - Type: bool
- X
    - Type: float

1.4.2 Branches

- Branch 1:
    - If the throttle is >= than 0.25 OR <= than -0.25, set brake sound played to false. Otherwise, continue to branch 2.
- Branch 2:
    - If the absolute value of x is greater than or equal to 10.0, continue to branch 3. Otherwise, the program is terminated.
- Branch 3:
    - If a brake sound is played, the program is terminated. Otherwise, the Play Sound 2D function is called.

1.4.3 Functions

- Play Sound 2D
    - Arguments: sound, volume multiplier = 1.0, pitch multiplier = 1.0, start time = 0.0, concurrency settings, owning actor, Is UISound = true
    - Explanation: sets brake sound played to true

2. Dirt Road Sound (written by Ian Yew)

Dirt Road Sound.png

2.1 Variables

- Road sound
- Road rough
    - Type: float
- RPM
    - Type: float

2.2 Functions

- Set Float Parameter
    - Takes in a float value and sets road sound to the particular float value

3. Audio (written by Ian Yew)

Audio.png

3.1 Variables

- Engine Sound
    - Type: audio
- RPM
    - Type: float

3.2 Functions

- Set float parameter
    - Takes in float value, sets engine sound to the float value

4. Handle mouse interior camera movement (written by Ian Yew)

Handle mouse interior camera movement

4.1 Variables

4.2 Functions

- GetLookUp
    - Returns float value
- Break rotator
    - Returns X(roll), Y(pitch), Z(yaw), takes in relative rotation detected from internal camera
- GetLookRight
    - Returns float value
- Clamp Angle
    - Takes in angle degrees = GetLookUp() + BreakRotator(rotation).Y, Min Angle Degrees and Max Angle Degrees of -89.0 and 89.0 respectively
    - Returns a float value to be used in the Make Rotator function
- Make Rotator
    - Arguments: X(Roll) = 0.0, Y(pitch) = ClampAngle's return value, Z(yaw) = Break Rotator(Rotation).Z + GetLookRight()
    - Returns float value to be used in Set Relative Rotation function
- SetRelativeRotation
    - Arguments: Internal Camera(scene component), New Rotation = MakeRotator's return value
    - Sets internal camera's relative rotation to the New Rotation value

5. Collisions (written by William Chen)

Collisions

5.1 Main Goal

- Detect what has been hit by the truck and play a sound feedback from collision

5.2 Variables

- Collision Sound
    - Explanation: a variable to hold how loud the volume of the sound file should be when played
    - Type: Float
    - Range: 0 to maximum float number (realistically will never reach this high)

5.3 Functions

- Event ActorBeginOverlap
    - This event is triggered when an actor overlaps with another actor (e.g., a vehicle collides with a tree).
    - **Other Actor** - what is being collided into (e.g., tree from previous bullet point)
    - The "***Print String***" function that this connects to is an **OPTIONAL** logic. You can delete this entirely, and the code will still run fine. The purpose of using this is so there will be a debug print message on the screen during the simulation to see what the name of the actor that has been hit is called.
- Get Forward Speed
    - Takes the input "Simple Wheeled Vehicle Movement" which is a built-in component on vehicle objects. You can find this in the upper left where it lists the components of the actor.
    - Multiply by 0.036 and then divide by 45 and then ABS (take the absolute value), the numbers determined here are arbitrary and they are used to adjust the volume of the sound that comes from "**Play Sound 2D".** The faster the vehicle drives, the louder the sound will be.
    - This gets set to float variable "***Collision Sound***"
- Get Display Name
    - *Check what the name of "**Other Actor**" is.*
- Switch on String
    - Based on the name returned from "**Get Display Name**", find the case that matches exactly with that returned name (case sensitive).
    - **InstanceFoliageActor -** this is the name of certain objects in the world, this can include trees, bushes, etc. (not an exhaustive list of everything, but those are the main objects with that name)

    - **StartCurve -** this is the name of certain objects in the world, this can include guard rails, buildings, etc. (not an exhaustive list of everything, but those are the main objects with that name)
    - **Default -** this is the “else” case where none of the names match up
- ***Play Sound 2D***
    - Depending on which case from “**Switch on String**” is returned, play a certain sound file

5. Camera Zoom (written by Sooah Park)

Camera Zoom

1. **Main Goal**

- Enabling zoom in/out with mouse wheel control

2. Variables

- Target Arm Length
    - Explanation: Variable of the Spring Arm object which defines the distance between the camera and the origin of the Spring Arm (or player/character)
    - Type: Float
    - Values: Range of values from either 150-1500 or 0-1500

3. Other objects

- Mouse Wheel Up/Down
    - Explanation:
        - Up: When mouse wheel is scrolled up, move to SET function which will set the Spring Arm according to clamp value between 150 to 1500
        - Down: When mouse wheel is scrolled down, move to SET function which will set the Spring Arm according to clamp value between 0 to 1500
- Spring Arm
    - Explanation: Camera position; the distance between the camera and the player (adjusted by the Target Arm Length)

4. Functions

- SET
    - Set the target (Spring Arm) to a value of Target Arm Length multiplied either by 0.91 (mouse wheel up) or 1.1 (mouse wheel down)
- Clamp (float)
    - Controls the value of Target Arm Length/Spring Arm (min/max), between 150-1500 for mouse wheel up and 0-1500 for mouse wheel down
- x (“Multiply” Functions)
    - If Mouse Wheel Up: Multiplies 0.91 to the target arm length and returns the result
    - If Mouse Wheel Down: Multiplies 1.1 to the target arm length and returns the result

6. Initialization (written by William Chen)

Initialization

1. Main goal

- When simulation starts, calibrate the vehicle and steering wheel

2. Variables

3. Functions

Event BeginPlay

- Explanation:
  - Every blueprint will always have this event. This event is called every time the simulation is played for one frame.

Cast To MT_AnimBP

- Explanation:
  - Casting allows a blueprint to take a reference of another class. This means you will be able to call variables or functions from that other class.
- Note that this call is very expensive in memory, so frequent usage of "Cast To" functions may result in crashes or lag.

Camera Switch

Initialize Wheel (Custom Function from Logitech Library)

- Explanation:
  - Calibrates steering wheel (the spinning that happens on the wheel)
- Parameters:
  - **Ignore XInput Controllers**: if set to true, the SDKs will ignore any X-input controller (X-input means Xbox controllers)

Delay

- Explanation:
  - Waits X seconds before continuing to the next function (e.g. In our case, we have it wait 1 second).
- Reason:
  - We do this because we want to ensure that the wheel is initialized before any wheel function has a chance to happen. If we do not do this, then wheel functions might not work because initialization has not happened yet.

Print String

- **OPTIONAL** logic. You can delete this entirely, and the code will still run fine. The purpose of using this is so there will be a debug print message on the screen during the simulation to see whether the wheel was

properly initialized. **If it ever prints FALSE, it means the steering wheel will not work.**

-

Play Wheel Spring Force (Custom Function from Logitech Library)

- Parameters:
    - **(int) Index**: how to tell what device/controller is plugged in (e.g. 0 = first controller, 1 = second controller, etc.)
    - **(int) Offset Percentage [-100, 100]:** Specify the center of the spring force effect. Valid range is -100 to 100. Specifying 0 centers the spring. Any values outside this range are silently clamped.
    - **(int) Saturation Percentage [0, 100]:** Specify the level of saturation of the spring force effect. The saturation stays constant after a certain deflection from the center of the spring. It is comparable to a magnitude. Valid ranges are 0 to 100. Any value higher than 100 is silently clamped.
    - **(int) Coefficient Percentage [-100, 100]:** Specify the slope of the effect strength increase relative to the amount of deflection from the center of the condition. Higher values mean that the saturation level is reached sooner. Valid ranges are -100 to 100. Any value outside the valid range is silently clamped.

Get All Actors Of Class (Target Laser)

- Get all **laser type** objects
- **SET:**
    - Any object that are laser types gets set to an array that holds those laser objects. Allows for quick calls to lasers when needed through loops.

Get All Actors Of Class (Data Collector Ref)

- Get all **data collection** type objects
- **SET:**
    - Any object that is data collection types gets set to a single Datacollector variable (of type Data Collection) that holds those data collection objects. Allows for quick calls to data collection when needed through loops.
    - We only get the first index (0) from the data collection type array because we only want access to 1 “data collector” to collect data.

Set Timer by Event

- Set a timer to execute

- Parameter:
    - **Event:** what should trigger this timer
    - **Time (float):** how long should this timer run for
    - **Looping (boolean):** should this event repeat
    - **Initial Starting Delay (float):** how much time delay (in seconds) before the timer runs
    - **Initial Starting Delay Variance (float):** add some variance to when the timer starts (in seconds) in lieu of doing a random range on the InitialStartDelay input
- We use these events so that custom events will run continuously (hence looping is checked to be true)

7. Handle Mouse Exterior Camera Movement (written by Ian Yew)

Handle mouse exterior camera movement

1. **Main goal**
    - Setting relative rotation according to camera movement
2. **Variables**
    - Rotation of spring arm
3. **Functions**
    - GetLookUp
    - Break Rotator
        i. Takes in relative rotation, and return values of x(roll), y(pitch) and z(yaw)
    - GetLookRight
    - ClampAngle
        i. Takes in the sum of Y(pitch) from Break Rotator function and GetLookUp, with set min and max angle degrees to return a value
    - Make Rotator
        i. Takes in a default x(roll) value, Y(pitch) from ClampAngle and Z(yaw) from summing Z(yaw) from break rotator and value from GetLookRight
        ii. Returns the value of the new rotation

8. Axis Input (written by Sooah Park)

Axis Input

1. **Main Goal**

- Reading user input to apply it to the vehicle movement

2. **Variables**

- Throttle
    - Explanation: acceleration value applied to the vehicle
    - Type: Float
- Throttle Data
    - Explanation: temporary value used to calculate the throttle value
    - Type: Float
- Brake Data
    - Explanation: brake force value applied to the vehicle
    - Type: Float
- Reverse
    - Explanation: if the vehicle is in reverse gear
    - Type: Boolean
- Steer
    - Explanation: steering value applied to the vehicle
    - Type: Float
- Steer Angle
    - Explanation: value passed into each wheel
    - Type: Float

3. **Other objects**

- InputAxis MoveRight
    - Captures left/right movement from steering input and passes it to the Steer variable
- InputAxis MoveForward
    - Captures forward/backward movement from input, sets Throttle Data and Brake Data variables based on the input

4. **Functions**

- GetWorld Delta Seconds
- FInterp To
    - Changes the input value (Throttle Data/Brake Data) gradually (with the SET Throttle Input/Brake Input functions)
- Set Steer Angle

- Sets steering angle to a specific wheel with the input Steer Angle and Wheel Index

9. Send RPM and Speed to AnimBP (written by Sooah Park)