VR Driving Simulator Script Documentation

Contents

DataSave

Purpose

DataSave saves certain simulator data frame-by-frame and exports to a text file in CSV format unique to each test run.

Overview

- This script is to be attached directly to the Participant object.
- At the start of simulation, DataSave creates a new text file based on current date and time, numerically formatted as “YearMonthDay_HoursMinutesSeconds.txt”
- At every frame, DataSave pulls data from vehicle components and writes it in .CSV format to a text file at a given file path. Data is taken from LogitechSteeringWheel, unityDrivingLatency, and RigidBody.
- Important data includes:
    - Frame index
    - From unityDrivingLatency:
        - Current speed
        - Current acceleration
        - Angular velocity
        - Pitch
        - Roll
        - Yaw
    - From LogitechSteeringWheel
        - Physical angle of wheel
        - Physical angle of brake pedal
        - Physical angle of gas pedal
        - If reverse gear is on
        - If neutral gear is on
    - From RigidBody
        - Current X-position and X-angle
        - Current Y-position and Y-angle
        - Current Z-position and Z-angle

Important Variables

- Filepath
    - Public string, determines folder to save all text files
    - Determined before runtime in Inspector
- Filename
    - Public string, determining name of text file
    - Initialized during runtime based on current date and time

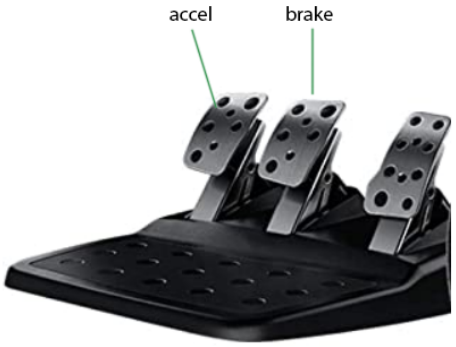
LogitechSteeringWheel

Purpose

LogitechSteeringWheel is the interface between the physical Logitech equipment and the simulator car model. It translates actions such as turning the wheel or pressing a pedal into user input, which is then referenced by unityDrivingLatency.

Overview

- This script is to be attached directly to the Participant prefab object.
- LogitechSteeringWheel checks whether a Logitech GSDK is connected to determine whether to receive input from Logitech equipment or keyboard (it is possible, but not recommended, to receive input from both sources)
    - If the keyboard is active, “C” controls the gas, “X” controls the brake, and the mouse controls the wheel. Up and down arrow keys used to switch gears.
    - If keyboard is inactive and Logitech is connected, the Logitech Steering Wheel, Gear Shift, and Pedals will control respective inputs
        - Due to unknown bug, gas is currently handled by clutch pedal (to left of brake)
        - On the gear shift, all downward shifts are drive, all middle are neutral, top-rightmost gear is park, and the rest are reverse.
- At every frame, LogitechSteeringWheel records the current state of inputs from the GSDK and stores them in variables:
    - All button and paddle variables are booleans
    - Accel, brake, and wheel are floats measuring the angle said pedal/wheel is pushed/turned.
        - Accel range [0, 1]
        - Brake range [0, 2]
        - Wheel range [-1, 1]



- Wheel button functions
    - aButton - turn on/off Cruise Control
    - bButton - turn on/off Right Blinker
    - xButton - turn on/off Left Blinker
    - yButton - turn on/off Hazard Lights
    - leftPaddle - decrease cruise speed (when cruise is active and speed is LOCKED)
    - rightPaddle - lock cruise speed when cruise is active/increase cruise speed
    - button6 - currently unused
    - button7 - currently unused
    - button8 - currently unused
    - button9 - currently unused
- Pedal functions
    - Accel - increase car acceleration
    - Brake - increase brake torque on all wheels, slowing car down
- Gear shift functions
    - Logitech gear shift has 9 shiftable gears, arranged in a 3x3 square grid.
        - Bottom three gear positions are stored as bools button13, button15, and button17 from left to right. When in these positions, car is in DRIVE gear.
        - Top left and top middle positions (button12 and button14) put car in REVERSE gear.
        - Top right position (button16) puts the car in PARK gear.
            - PARK can only be engaged while brakes are being applied, and subsequently can also only be disengaged while brakes are applied.
        - Middle three gear positions represent NEUTRAL gear. All 3 positions are represented by the bool “neutral”, which returns true when all of buttons 12 through 17 are false.
    - Depending on which gear is active, the corresponding letter on car dashboard will light up. (D - drive, N - neutral, R - reverse, P - park)
- Frame-by-frame algorithms
    - Parking gear
        - When user shifts into park, code checks if brakes are being applied.
            - If brakes are applied, park is engaged.
            - If not, car will default to neutral and user will have to shift back out of park position.
        - Once in park, code checks if brake is active when shifting out:
            - If brakes are applied, gear successfully changes
            - If not, car will remain in park and user must shift back to park, then apply brakes while shifting out to properly leave.
    - Cruise control
        - Enabled/disabled with aButton
            - Can only be turned on when over 25 MPH
            - When enabled, the blue cruise control symbol lights on dashboard

        - Once symbol is visible, user can accelerate to desired speed and lock a target speed by pulling the right paddle shift.
            - Locking speed will display target speed on the dashboard.
            - Pressing aButton again will disable cruise control
        - While the speed is locked, user can increase/decrease the target speed by using the right and left paddle shifts respectively.
            - Once speed is locked, cruise control can be disabled by either pressing aButton, applying the brakes, or pressing on the gas.
- LogitechSteeringWheel checks the type of connected gear shift and stores as a string
    - Sequential: Only up/down movements between gears
    - Gated: gear movements go up/down/left/right.
        - Logitech uses gated shifting for the simulator
- LogitechSteeringWheel determines spring force on the car based off of current speed and constants surfaceMag and surfaceFreq
    - Utilizes Logitech GSDK DLL import from LogitechSteeringWheelEnginesWrapper.dll
    -

Important Variables

- surfaceFX
    - public bool determining whether to apply additional spring forces depending on surface
- Wheel, accel, brake
    - Public floats measuring angle of pedal press and wheel turning.
    - Stored by DataSave in test data
- Neutral, park, drive, reverse
    - Public bools that individually yield true when their respective gear is active
- Cruise
    - Public bool telling unityDrivingLatency to switch control of acceleration from SteeringWheel.accel to an algorithm that adjusts acceleration depending on difference between current and target speed.
- Keyboard
    - Public bool manually set in Inspector. Determines whether to check Logitech equipment for input or keyboard presses.
- currentGear
    - Public int determining which gear is currently active
        - 0 - park
        - 1 - reverse
        - 2 - neutral
        - 3 - drive
    - Calculated in updateGear() and referenced by GearDisplay in UpdateGear(), called within Update() of LogitechSteeringWheel

Important Constants

- For determining force applied from springs to vehicle:
    - springCenter (default 0)

    - springMagnitude (default 80)
    - springFalloff (default 80)
- For determining surface effect
    - surfaceFX (default False), determines whether to apply surface effects
    - surfaceMagMin (default 5)
    - surfaceMagDelta (default 15)
    - surfaceFreqMin (default 70)
    - surfaceFreqDelta (default -10)

Important Functions

- void checkExtremes(LogitechGSDK.DIJOYSTATE2ENGINES rec)
    - The function checkExtremes() is a debug function that checks whether the current wheel rotation is the minimum or maximum of the current wheel range in context of the test run
    - Stores current rotation in maxx or minx depending on if it is the smallest or largest angle of the wheel based on current test run
- void updateGear()
    - The function updateGear() checks which gear bool is currently active and passes a corresponding int to the variable currentGear

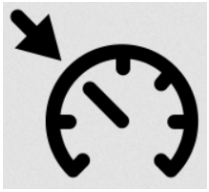
unityDrivingLatency

Purpose

unityDrivingLatency takes user inputs from LogitechSteeringWheel such as accel and brake, combined with Rigidbody properties and set constants to generate test debug and output variables to control motor direction.

Overview

- This script is to be attached directly to the Participant prefab object.
- unityDrivingLatency first applies the default configuration to all wheel colliders
    - All vehicle substeps defaulted to 10
- Position of Virtual Reality Input Tracking camera position is recentered to (0, 0, 0) on the center of the driverPos object (double check this)
- Inputs are parsed from LogitechSteeringWheel to initialize important booleans
    - Reverse and neutral are bools that set to true or false when respective floats are at their max or min respectively
- Depending on if cruise control is active, value for float acceleration is determined either by LogitechSteeringWheel.accel or by cruise control algorithm.
    - Enabled/disabled with aButton
        - Can only be turned on when over 25 MPH
        - When enabled, the blue cruise control symbol lights on dashboard
    - Once symbol is visible, user can accelerate to desired speed and lock a target speed by pulling the right paddle shift.
        - Locking speed will display target speed on the dashboard.
        - Pressing aButton again will disable cruise control
    - While the speed is locked, user can increase/decrease the target speed by using the right and left paddle shifts respectively.
        - Once speed is locked, cruise control can be disabled by either pressing aButton, applying the brakes, or pressing on the gas.



- Value of brake float is determined by multiplying the current LogitechSteeringWheel.brake value by a clamped range [1, 10] of current speed, divided by 10.
    - When current speed is over 10, the full LogitechSteeringWheel.brake value is applied
    - Once speed drops below 10, the strength of the brake is reduced from 100% to a minimum of 10% once current speed hits 1 MPH.
- unityDrivingLatency calls the function Move() in CarController
    - Passes user input values for wheel, brake, acceleration, and checks whether car is in Reverse or Park gear.
    - wheelTest and brakeTest (defaults 0) are added to values for wheel and brake as a means for debugging and testing.
        - Values for wheel and brake themselves cannot be changed as they are updated every frame by LogitechSteeringWheel.
- Physics.Simulate() is ran at rate inversely dependent on fixed target frame rate (default 90, runs every 1/90th second)

- Pitch, roll, and yaw describe the state of the vehicle with respect to its current speed, acceleration, angular velocities adjusted by multiple other factors
    - Pitch is calculated from the z-acceleration multiplied with a function of brake to avoid strong forward pitch upon braking.
        - Change in pitch is smoothened by comparing current value of pitch to previous pitch values.
        - It is then filtered with constant parameters (pitch_lp, lp_lim, pitch_hp) and adjusted for max speed and bumpiness generated by Random.range() before being multiplied by a constant
        - Final pitch is clamped between (-70, 70), can be changed
    - Roll is calculated from x-acceleration, angular velocity, and current z-speed.
        - Change in roll is smoothened by comparing current roll value to previous roll values
        - Roll is filtered further with constant parameters (roll_lp, lp_lim, roll_hp) and adjusted for current speed fraction and bumpiness from Random.range() before being multiplied by a constant
        - Final roll is clamped between (-120, 120),can be changed
    - Yaw is calculated with angular velocity
        - Yaw is smoothened out based on changes from previous yaw values
        - Yaw is further adjusted by yaw_hp (default 0) before being washed out.
        - Final yaw is clamped between (-3200, 3200), can be changed, but not recommended as Yaw currently bears little to no impact on the state of the motors
- Car Dashboard UI is updated based on current values
    - Speed display is displayed as current speed multiplied by 2.23693629.
    - Steering Wheel object is rotated with respect to the rotation of the Logitech Steering Wheel.
    - Speedometer rotates the needle (when enabled) based on the fraction of the current speed and maximum speed
    - Tachometer (when enabled) displays calculated estimated revolutions per minute of the engine, indicated by CarController.revs
- Telemetry is updated and a package is sent to SimRacingStudios script containing essential variables for motors
    - Name, API, API version, vehicle, and game name of package
    - Value of pitch multiplied by constant scale
    - Value of roll multiplied by constant scale
    - Value of yaw multiplied by constant scale
    - Several numeric values defaulted to 0 with minimal bearing on simulator performance

Important Variables

- acceleration
    - Final calculated acceleration depending on either gas pedal or cruise control algorithm
- brake

    - Final calculated brake value based off of current speed
- pitch_result
    - Final calculated pitch value describing vehicle's physical rotation along the y-axis. This is sent to SimRacingStudio script to adjust motor controlling pitch
- Roll_result
    - Final calculated roll value describing vehicle's physical rotation along the x-axis. This is sent to SimRacingStudio script to adjust motor controlling roll
- Yaw_result
    - Final calculated yaw value describing vehicle's physical rotation along the z-axis. This is sent to SimRacingStudio script to adjust motor controlling yaw.
- currentSpeed_z
    - Measures the current forward speed of vehicle. This is sent over to DataSave for data collection.
- currentAccel_z
    - Measures current forward acceleration of vehicle. This is sent over to DataSave for data collection.
- angularAccel
    - Measures current angular acceleration of vehicle. This is sent over to DataSave for data collection.

Important Constants

- defaultAccel
    - Determines default acceleration when drive gear is engaged by gas pedal is inactive
- pitchTest
    - Public float that can be altered in Inspector outside and during Play Mode to test impact of pitch values with motors.
- rollTest
    - Public float that can be altered in Inspector outside and during Play Mode to test impact of roll values with motors.
- yawTest
    - Public float that can be altered in Inspector outside and during Play Mode to test impact of yaw values with motors.

CarController

Purpose

CarController is responsible for performing all operations needed for control of the vehicle. This includes steering front wheels; applying brake torque to rear wheels; traction, spin, and skid control; and overall application of steering, gas, brake, and current gear on car motion using Wheel Colliders.

Overview

- This script is to be attached directly to the Participant prefab object.
- This script does not contain an Update() function, and instead is a collection of essential functions that are called internally and by unityDrivingLatency to handle car movement from the four wheel colliders attached.
- Upon start, local rotations and center-of-mass of each of four wheel colliders and meshes are initialized.
- Current torque and rigidbody components are initialized.

Important Functions

- private void GearChanging()
    - The GearChanging() function checks vehicle's current speed as fraction of max speed and sets a gear number (1-10) with respect to that fraction.
        - Higher fraction = higher gear number.
    - Called by both Move() functions
- private static float CurveFactor(float factor)
    - The CurveFactor() function takes an input x and passes it through the function f(x)= 1 - (1-x)(1-x), yielding a curved bias towards 1 for a value in the (0, 1) range.
    - Called in CalculateRevs()
    - Parameters:
        - Float factor - a float ideally between 0 and 1.
- private static float ULerp(float from, float to, float value)
    - The ULerp() function acts similar to the built-in Lerp function, but allows the yield to exceed the from-to range
    - Given floats "from", "to", and "value", the function returns a float in the form of f(x)= (1 - value)*from + (value * to)
        - Mathf.Lerp() normally returns the interpolated float result between the "from" and "to" floats by "value" variable.
    - Called in CalculateRevs()
- private move Move(float steering, float accel, float footbrake, float handbrake, float reverse, float park)
    - The Move() function takes user input values from LogitechSteeringWheel such as steering (Logitech.wheel) and park gear (Logitech.park), and calculated values from unityDrivingLatency such as acceleration, brake, and reverse.
    - The rotations and positions of each of the four wheel colliders are stored and assigned to the transforms of the four wheel meshes.
    - Steering input is clamped between [-1, 1] to avoid undesired input formats

    - Move() checks whether the car is in park to determine final calculated acceleration
        - If car is in Park, acceleration is exponentially decreased until speed drops below 3, at which the RigidBody velocity is frozen at zero.
    - Footbrake and handbrake are clamped between [0, 1]
        - It should be noted that input for handbrake is currently hardcoded as zero.
    - The final values for brake and accel are passed into ApplyDrive()
    - Values for steering wheel are applied to rotation of front wheel meshes, and values of brake are applied to the rear wheels.
    - SteerHelper(), CapSpeed(), CalculateRevs(), GearChanging(), AddDownForce(), CheckForWheelSpin(), and TractionControl() are all called in this function.
    - Called externally from unityDrivingLatency.cs
- Private void ApplyDrive(float accel, float footbrake)
    - Depending on the CarDriveType, motor torque is applied to the four wheel colliders based on value of accel.
        - FourWheelDrive - current torque is divided by 4 and applied to all four wheel colliders
        - FrontWheelDrive - current torque is divided by 2 and applied to only the front two wheels
        - RearWheelDrive - current torque is divided by 2 and applied to only the rear two wheels
    - Afterwards, brake torque is also apple to all four wheel colliders based on value of brake float.
    - Called in Move()
    - Parameters:
        - Accel - the final calculated value of acceleration as determined in Move()
        - Brake - final calculated value of brake determined in Move()
- public void Move(float steering, float accel, float footbrake, float handbrake)
    - An override of the Move() function, this function is identical to the first Move(), but lacks parameters and calculations to handle park and reverse gear.
    - This function is not called anywhere, since the Reverse and Park gears are crucial to the car's basic mechanics.
- private void CapSpeed()
    - The function CapSpeed() checks the current speed of the rigid body and sets specific cap multiplier based on the current speed type.
        - If Miles Per Hour, top speed is divided by 2.23693629
        - If Kilometers Per Hour, top speed is divided by 3.6.
    - Called from Move()
- private void SteerHelper()
    - The function SteerHelper() checks if wheels are on the ground and realigns the rigidbody velocity if so.
    - Calculations are made to avoid gimbal lock problems that would make the car suddenly shift directions.
    - Called in Move()

- private void AddDownForce()
    - The function AddDownForce() helps add more grip to the center wheel collider based on current velocity.
    - Called from Move()
- private void CheckForWheelSpin()
    - The function CheckForWheelSpin() checks if the vehicle's wheels are spinning and handles skid-related functions
        - Emits skid particles if spinning on ground
        - Plays skid sound effects if spinning on ground
        - Leaves skidmarks on ground if spinning on ground
    - Called in Move()
- private void TractionControl()
    - The function TractionControl() provides crude traction control that reduces the power to the wheel if the car is wheel-spinning too much.
    - Checks car type to determine which colliders should have torque adjusted on their forward slip with AdjustTorque()
        - If FourWheelDrive, all four wheels have torque adjusted
        - If FrontWheelDrive, only front two wheels have torque adjusted
        - If RearWheelDrive, only two rear wheels have torque adjusted
    - Called in Move()
- private void AdjustTorque(float forwardSlip)
    - The function AdjustTorque() takes a float and adjusts the torque value of wheel colliders based on forward slip
    - Called in TractionControl().
- private bool AnySkidSoundPlaying()
    - The function AnySkidSoundPlaying() checks if any of the four wheel colliders is currently playing a skid sound effect and returns true if any are.

Important Variables

- Skidding
    - Public bool returning true if the car is determined to be skidding
    - Currently unused
- m_SteerAngle
    - Private float calculated as (unityDrivingLatency.steering * m_MaximumSteerAngle) in Move()
    - Determines the angle at which to rotate the front wheels depending on rotation of steering wheel
    - Calculated and referenced in Move()
- m_GearNum
    - Public int calculated in GearChanging() to determine revs per minute based on current speed
    - Normally referenced in unityDrivingLatency in calculating drag
- m_GearFactor
    - Public float representing normalised representation of the current speed within the current gear's range of speeds.

    - Calculated in CalculateGearFactor() and referenced in CalculateRevs()
- m_OldRotation
    - Public float value representing previous y-angle of car, used to avoid gimbal lock problems
    - Calculated and referenced in SteerHelper()
- m_CurrentTorque
    - Private float measuring the current calculated torque value across the four wheel colliders
    - Calculated in AdjustTorque and referenced in ApplyDrive()
- Revs
    - Public float representing the revolutions per minute of the engine based on current gear, gear factor, and speed.
    - Referenced in unityDrivingLatency for updating the Tachometer UI
- AccelInput
    - Public float representing final calculated acceleration to be used for determining motor torque on wheel colliders
    - Calculated in Move() and referenced/passed to ApplyDrive()
- BrakeInput
    - Public float representing final calculated brake value to be directly used in determining brake torque on wheel colliders
    - Calculated in Move() and referenced/passed to ApplyDrive()

Important Constants

- m_MaximumSteerAngle
    - Public float representing the farthest angle the front wheels can rotate
    - Defaults at 50 degrees
    - Referenced in Move()
- m_SteerHelper
    - Private, serialized float constant ranging from [0, 1] that determines the amount of grip on the wheels
        - 0 is raw physics
        - 1 causes the car to grip in the direction it is facing
    - Referenced in SteerHelper()
- m_TractionControl
    - Private, serialized float constant ranging from [0, 1] that determines the amount of traction control
        - 0 is no traction control
        - 1 is full interference
    - Referenced in AdjustTorque()
- m_FullTorqueOverAllWheels
    - Private, serialized float constant that sets a maximum value for wheel torque if forward slip is not exceeding the slip limit (m_SlipLimit) or if there is current torque (m_CurrentTorque) is 0
    - Referenced in AdjustTorque()
- m_Downforce

    - Private float representing the constant amount of downforce applied on each of the four wheel colliders
    - Defaults at 100
    - Referenced in AddDownForce()
- m_Topspeed
    - Private float representing the maximum possible speed attainable by the Participant model
    - Defaults at 200
    - Referenced in CapSpeed()
- NoOfGears
    - Private, serialized static int constant measuring the number of gear partitions used for calculating current revolutions per minute based on current speed and max speed
    - Defaults at 10
    - Referenced in GearChanging(), CalculateGearFactor(), and CalculateRevs()
- m_RevRangeBoundary
    - Private, serialized float constant determining the speed boundary upon which current gear should be changed
    - Defaults at 1
    - Referenced in CalculateRevs()
- m_SlipLimit
    - Private, serialized float constant determining after how much slip is reached that skid effects should be applied
    - Defaults at 0.2
    - Referenced in CheckForWheelSpin() and AdjustTorque()
- m_BrakeTorque
    - Private, serialized float constant representing the default amount of brake torque for each of the four wheel colliders
    - Defaults at 10000
    - Referenced in ApplyDrive()

GearDisplay

Purpose

GearDisplay receives an int value from LogitechSteeringWheel indicating what the current gear is. It updates the current gear number and passes instructions to GearSwitch to appropriately update the Gear UI on the dashboard.

Overview

- This script is to be attached to the game object of the Participant prefab pertaining to the dashboard UI.
- This script does not run code per frame (no Update()), and is a collection of essential functions to be called internally and externally by LogitechSteeringWheel.cs to update the UI Dashboard with the current gear letter.

Important Functions

- public void UpdateGear(int gear)
    - The function UpdateGear() takes an int input [0, 3] and assigns it to currentGear before calling the function ChangeGear()
    - Called from Update() in LogitechSteeringWheel
- void ChangeGear()
    - The function ChangeGear() turns off all active gear UI symbols by calling each gear's TurnOff() function in GearSwitch.
    - The current gear symbol is then turned on by the TurnOn() function in GearSwitch

Important Variables

- public int currentGear
    - Public int representing the int value of the currently active gear
    - Starts as -1 to avoid conflicts
    - Referenced in ChangeGear() as the index of the gears GameObject array to be set active.
- public GameObject[] gears;
    - Public array of GameObjects initialized in the Inspector as containing the UI game objects for each of the gears' UI dashboard symbols
        - Gears[0] - contains symbol for park gear
        - Gears[1] - contains symbol for reverse gear
        - Gears[2] - contains symbol for neutral gear
        - Gears[3] - contains symbol for drive gear
    - Is used to iteratively disable all gear symbols in ChangeGear() before enabling the specific gear currently active

GearSwitch

Purpose

GearSwitch handles whether the gear symbol it is attached to should be active or inactive depending on the current gear value. It enables and disables certain child game objects of the UI symbol, making the symbol appear either lit or turned off.

Overview

- This script is to be attached to each of the four gear symbol objects present on the UI Dashboard.
    - Must be attached to the parent object, not any of the On/Off children
- This script does not contain an Update() function, and instead is a collection of essential functions that are called externally by GearDisplay to handle the on/off appearance of gear symbols on the dashboard.
- This script will have two primary children objects, representing the on/off states of the symbol, that are to be laid directly atop on another and must have matching size and shape
- At the very start, all instances of this script set the appearance of symbol to Off
    - When symbol is Off, the gearOn object is inactive and gearOff object is active
    - When symbol is On, the script switches which objects are active

Important Functions

- public void TurnOn()
    - The function TurnOn() sets the symbol's appearance to be "on"
        - gearOn object is set active
        - gearOff object is set inactive
        - Bool isOn is set to true
    - Called from ChangeGear() in GearSwitch
- public void TurnOff()
    - The function TurnOff() sets the symbol's appearance to be "off"
        - gearOn object is set inactive
        - gearOff object is set active
        - Bool isOn is set to false
    - Called from ChangeGear() in GearSwitch

Important Variables

- public GameObject gearOn
    - Public Game Object initialized in the Inspector
        - Dragging correspondingly-named child object to slot
    - Contains a collection of objects appearing as the first letter of gear name, with added glow and light effects, giving the appearance of a lit symbol
    - Is set active when the particular gear this script is attached to is told to activate by GearDisplay
- public GameObject gearOff
    - Public Game Object initialized in the Inspector
        - Dragging correspondingly-named child object to slot

    - Contains a collection of objects appearing as the first letter of gear name, with dim border and darkened internal color, giving appearance of the symbol turned off
    - Is set active when the particular gear this script is attached to is told to deactivate by GearDisplay
- public bool isOn
    - Public boolean checking whether this specific gear is on or off
    - Referenced in ChangeGear() in GearDisplay