SILANTRO FLIGHT SIMULATOR

SCRIPT DOCUMENTATION

Thank you for purchasing this asset, please don't forget to rate it on the asset store. This document contains a description of the variables, properties and methods contained in each script.

If you have any questions or unclear areas, please email me: **silantrosimulator@gmail.com**

SILANTRO CORE

`Oyedoyin.Common.SilantroCore: UnityEngine.MonoBehaviour`

This component is just as the name says 😊 it's right at the centre of the system functionality and performs a set of important functions;

1. Collects raw data about the aircraft/helicopter behavior and performance and filters them into forms required by the components.
2. It computes and sets the center of gravity of the aircraft/helicopter depending on the current weight and position of the individual (COG affecting) components
3. It also sets the inertia values of the base rigidbody if saddled with that responsibility.

PROPERTIES

Property:	**Function:**
`public SystemType functionality`	Determines if the aircraft/helicopter center of gravity is computed each time step or just based on the set empty cog gameObject position
`SystemType.Basic`	*Uses the local position of the assigned empty COG gameObject as the center of gravity of the aircraft/helicopter*
`SystemType.Advanced`	*Computes the aircraft/helicopter center of gravity based on the position and weights of the fuel tanks, payload and munitions*
`public TensorMode tensorMode`	Determines if the aircraft/helicopter rigidbody inertia tensor matrix (Unity only considers the principal axis) is automatically calculated by Unity from the collider sizes and position or is set manually by the user.
`TensorMode.Automatic`	*Uses the tensor matrix calculated by Unity*
`TensorMode.Manual`	*Uses the Pitch, Roll and Yaw Inertia values supplied by the user and*
`public SpeedType speedType`	NB: This is only required for aircraft models. It is uses to activate speed particle effects based on the speed of the aircraft.
`SpeedType.Subsonic`	*Maximum aircraft speed is less than Mach 1*
`SpeedType.Supersonic`	*Maximum aircraft speed is greater than Mach 1*

`public Rigidbody sampleAircraft`	This enables the script to examine the pre-calculated inertia tensor values and enable to user to properly determine/guess the needed tensor values (If the real life value is not available)
`public Transform emptyCenterOfMass`	This is the transform mentioned in the `SystemType` definition. This is used to mark the center of gravity of the aircraft/helicopter when it's empty. Further reading: http://avstop.com/ac/weightbalance/ch3.html
`public LayerMask groundLayer`	This is used to determine the objects to measure ground positions from i.e. What object(s) in your scene can be considered as ground. The position measure from them is used to calculate ground effect on the rotors and aerofoils.
`public float Ixx`	This is the aircraft/helicopter Rolling Moment of Inertia (kg/m^2) and is the property of the aircraft/helicopter that determines how the mass is distributed along the Z-axis which tends to resist rolling. Higher values require larger moment values to roll the aircraft/helicopter
`public float Iyy`	This is the aircraft/helicopter Pitching Moment of Inertia (kg/m^2) and is the property of the aircraft/helicopter that determines how the mass is distributed along the X-axis which tends to resist pitching motion.
`public float Izz`	This is the aircraft/helicopter Yawing Moment of Inertia (kg/m^2) and is the property of the aircraft/helicopter that determines how the mass is distributed along the Y-axis which tends to resist yawing motion.
`public float maxAngularSpeed`	This is the maximum allowable angular speed of the aircraft/helicopter in degrees per second. This helps to limit the aircraft angular instability and improve FLCS controls.
`public float rotationDrag`	This is a measure of the Unity angular motion drag and is used mostly on helicopter models to make them easier to control. Increasing the angular drag will help to slow down and balance the helicopter when cyclic inputs are supplied.

`public AudioClip sonicBoom`	This is the audio played when the aircraft speed exceeds Mach 1 to give feedback to the user about the transition process
`public ParticleSystem sonicCone;`	This particle effect is used in conjunction with the audio clip above to simulate transition effects at high speeds on the aircraft
`public float sonicEmission`	Maximum level of emission of the sonic cone particle effect

INTERNAL FUNCTIONS

INITIALIZE

Creates and configures the required transforms, sets the rigidbody inertia values, configures the atmosphere component, sets the rigidbody angular drag and speed limits.

COMPUTE (`double dt`)

Receives computation command from the controllers scripts and calls the in script compute functions

`double dt: This is the fixed timestep value from the controller. This value is sent from the controller to give the user max control over the integration/interpolation timestep.`

COMPUTEDATA (`double _timestep`)

Collects linear, angular speeds from the rigidbody component. Filters them and calculates the linear and angular accelerations.

COMPUTECG

Calculates the effect of the fuel tank, payload, munition and gun drum weight and position on the rigidbody center of gravity.

SILANTRO TANK

`Oyedoyin.Common.SilantroTank: UnityEngine.MonoBehaviour`

The fuel tanks are designed to be an independent components that can be added or removed from the model hierarchy as the user pleases. It contains information about the current amount of fuel in a particular location within the model.

The fuel tanks can either be **internal** or **external.** External tanks can have a model attached to it and can be detached from the aircraft.

PROPERTIES

Property:	**Function:**
`public TankType tankType`	Determines if the fuel tank is within the aircraft structure or outside of it. This helps to check if the tank can be detached/jettisoned or not.
`TankType.Internal`	*Indicates that the tank is located inside the aircraft structure and cannot be detached at runtime*
`TankType.External`	*Indicates that the tank is located outside the aircraft structure and can be detached at runtime.*
`public TankPosition tankPosition`	This indicated the position of the aircraft relative to the zero/center point. This helps to determine what order fuel will be used from the tanks to keep the center of gravity within limits.
`public Mode mode`	This is an indication of the design mode the user wants to employ for the fuel tank.
`Mode.Volume`	*The x, y and z values of the local scale of the tank will be used to determine the dimensions of the fuel tank which will then be used the calculate the maximum volume of the tank.* *The fuel weight will then be calculated from the volume value and the density of the selected fuel.*
`Mode.Static`	*The user can input the maximum capacity of the fuel tank.*

`public FuelUnit fuelUnit`	This allows users to input or calculate the total capacity of the tank in non-metric units which will then be converted into kilograms
`public FuelType fuelType`	The Jet or AVGas fuel selected for the tank
`public float m_rightTaper`	Helps to shape the wing fuel tanks by adjusting the taper on the right side of the tank.
`public float m_leftTaper`	Helps to shape the wing fuel tanks by adjusting the taper on the left side of the tank.
`public float m_fillLevel`	Determines the percentage of the maximum capacity of the tank loaded.

Internal Functions

Initialize

Call the convert fuel functions and store the fuel tank capacity value

ConvertFuel

Convert the fuel capacity from the input value and the selected unit to kilograms

SILANTRO ACTUATOR

`Oyedoyin.Common.SilantroActuator: UnityEngine.MonoBehaviour`

The actuator component is used to drive movable parts on the helicopter/aircraft based on the imported animation sequence and a supplied input value (actuation level)

PROPERTIES

Property:	**Function:**
`public ActuatorType actuatorType`	Specifies the surface (type of surface) to be driven by the actuator component.
`public ActuatorState actuatorState`	Specifies the current state of the actuator
`public ActuatorMode actuatorMode`	Specifies the initialization sate of the actuator. E.g The gear actuator should be Open on start while a Speed brake actuator should be closed on start
`public SoundType soundType`	Specifies the logic that will be used for the actuator sound system
`SoundType.Simple`	*This will play the two specified audio clips when the Engage and Disengage functions are called using the PlayOneShot call*
`SoundType.Complex`	*This is play the loop Audioclip when the actuator is in motion and play the end clip when the actuator reaches the specified target point*
`public AudioClip EngageClip`	This is the audio clip to be played once when the Engage actuator function is called.
`public AudioClip DisengageClip`	This is the audio clip to be played once when the Disengage actuator function is called
`public AudioClip EngageLoopClip`	This is the audio clip to be played while the actuator component is in motion e.g What sound the door makes while its opening or closing
`public AudioClip EngageEndClip`	This is the audio clip to be played once when the actuator reaches the target point.

`public Animator actuatorAnimator`	The animator component containing the animation clip and animation controller.
`public string animationName`	String name of the animation clip.
`public int animationLayer`	Specifies the layer the animation clip is on in the Animation controller component.
`public bool generatesDrag`	Specifies if the movement of the actuator affects the aerodynamic drag of the aircraft/helicopter. Components like gears, speed brakes, doors etc.
`public float dragFactor`	This is the maximum aerodynamic drag coefficient when the actuator is in the open position.
`public bool invertMotion`	Specifies if the animation should be played backwards
`public bool invertDrag`	Specifies if the drag should reach its maximum value when the actuator is closed.

INTERNAL FUNCTIONS

INITIALIZE

Configures the animator components and creates the audio source components to be used by the assigned audio clips.

ANALYSESTATE (`int set`)

This functions determines and sets the current state of the actuator based on the input state target (`int set`) value.

`int set: This is the state target value which determines if the state of the actuator should be engaged or disengaged.`

`Set = 0: The actuator will be disengaged if it is closed on start`

`Set = 1: The actuator will be engaged if it is closed on start`

ANALYSESOUND (`float target`)

This functions determines what sounds should be played and in what order based on the input state target.

`float target = 0: The actuator is opening, so play the open/loop clip and follow the open sound path`

float target = 1: The actuator is closing, so play the close/loop clip and follow the close sound path

COMPUTE (double dt)

Receives computation command from the controllers scripts and calls the in script compute functions

double dt: This is the fixed timestep value from the controller. This value is sent from the controller to give the user max control over the integration/interpolation timestep.

CALL FUNCTIONS

ENGAGEACTUATOR

Commands the actuator to open, calculates the new drag value and plays the open sound effect

DISENGAGEACTUATOR

Commands the actuator to close, calculates the new drag value and plays the open sound effect.

SILANTRO AIRFOIL

`Oyedoyin.Common.SilantroAirfoil: UnityEngine.MonoBehaviour`

This component stores the aerodynamic properties, curves and variable required by the aerofoils and rotors. It is created from the main component menu with data from a Javafoil txt file or from an AFL file from.

PROPERTIES

Property:	Function:
`public string Identifier`	Specifies the name of the airfoil e.g. NACA 0013
`public float maximumThickness`	Describes the maximum thickness along the sides of the airfoil as a percentage of its chord length. Represents the (t/c) value from Javafoil
`public float thicknessLocation`	Describes the point along the chord length (in percentage) where the airfoil has its maximum thickness value
`public float leadingEdgeRadius`	Describes the radius of an imaginary circle drawn through the nose (leading edge) of the airfoil as a percentage of the chord length. Represents a measure of the roundness or bluntness of the airfoil leading edge.
`public float airfoilArea`	Describes the surface are of the 2D airfoil with a chord length of 1m.
`public AnimationCurve liftCurve`	This is a curve describing the relationship between the lift coefficient and the angle of attack in degrees.
`public AnimationCurve dragCurve`	This is a curve describing the relationship between the drag coefficient and the angle of attack in degrees.
`public AnimationCurve momentCurve`	This is a curve describing the relationship between the moment coefficient and the angle of attack in degrees.
`public float maxCd`	This is the maximum drag coefficient the airfoil will encounter during simulation. It is usually found at high post stall angles.

`public float upperStallAngle`	This is the airfoil stall angle in the positive angle of attack region.
`public float lowerStallAngle`	This is the airfoil stall angle in the negative angle of attack region.
`public float upperLiftLimit`	This is the maximum lift coefficient in the positive angle of attack region (lift coefficient at positive stall point)
`public float lowerLiftLimit`	This is the maximum lift coefficient in the negative angle of attack region (lift coefficient at negative stall point)
`public float zeroLiftAOA`	This is the angle of attack where the lift coefficient is 0.
`public float centerLiftSlope`	This is the slope of the lift curve calculated from the positive and negative maximum points.

Internal Functions

FilterPoints

This function calculates and separates the major airfoil properties from the curves and coefficient lists e.g. Maximum lift and drag coefficients, Upper and Lower stall points and the divergence Mach number

AnalyseCriticalMachNumber (`float sweepAngle, float cl`)

Calculates the critical Mach number of the airfoil based on the thickness, aerofoil sweep angle and current lift coefficient.

`float sweepAngle: The sweep angle of the aerofoil in degrees`

`float cl: The lift coefficient of the aerofoil panel/subdivision`

CheckThickness

This function calculates the leading edge radius of the airfoil and calls the FilterPoints () function

SILANTRO BULB

`Oyedoyin.Common.SilantroBulb: UnityEngine.MonoBehaviour`

This component controls the light component attached to it and the helicopter/aircraft by extension.

PROPERTIES

Property:	Function:
`public LightType lightType`	Specifies the mode of operation of the bulb. This determines what the operation curve will look like.
`public CurrentState state`	Specifies the current state of the light bulb, whether it's on or off
`public GameObject bulbCore`	This is the core light object with the illumination material assigned to it.
`public Light bulbLight`	This is the Unity light component attached to the light bulb.
`public Color bulbColor`	Specifies the color to display on the light component and core illumination material.
`public AnimationCurve flashCurve`	This curve is automatically generated based on the selected bulb type and describes the way the light illumination varies with time.
`public float blinkOffset`	This value is used to offset the bulb flash curve and properly coordinate the light displays or signals.
`public float maximumLightIntensity`	Specifies the maximum intensity to assign to the light component
`public float maximumEmission`	Specifies the maximum intensity of the core material illumination material
`public float blinkFrequency`	This is a measure of how fast the light bulb switches states i.e. how fast it blinks

Internal Functions

Initialize

Configures the bulb materials, sets the state and plots the flash/operation curve

PlotFlashCurve

Configures the operation curve based on the selected bulb type. It plots on a time scale, the points where the light intensity is at maximum and when it should be off.

Compute (`double dt`)

Receives computation command from the controller script and evaluates the current intensity of the light bulb from the flash curve.

`double dt: This is the fixed timestep value from the controller. This value is sent from the controller to give the user max control over the integration/interpolation timestep.`

Call Functions

SwitchOn

This is the function to turn on the Light bulb.

SwitchOff

This is the function to turn off the Light bulb.

SILANTRO CAMERA

`Oyedoyin.Common.SilantroCamera: UnityEngine.MonoBehaviour`

This component controls the interior and exterior cameras attached to the aircraft.

PROPERTIES

Property:	Function:
`public CameraFocus cameraFocus`	This will be used later to differentiate between VR and non-VR camera setups
`public CameraState cameraState`	Specifies the current state of the camera system i.e Interior or Exterior.
`public CameraMode cameraMode`	Specifies the functionality of the camera system, whether it will orbit around the aircraft (Orbit Mode) or can be rotated horizontally and vertically around the aircraft (Free Mode)
`public CameraStartState startState`	This specifies if the camera should start in the Exterior or Interior state.
`public CameraAttachment attachment`	Specifies if there are objects that will be enabled or disabled based on the current camera state e.g. pilot body model
`public GameObject exteriorObject`	This object will be active when the camera is in exterior mode and will be disabled when the camera switches to interior mode e.g. pilot 3d model
`public GameObject interiorObject`	This object will be active when the camera is in interior mode and will be disabled when the camera switches to exterior mode
`public Camera normalExterior`	This is the non VR exterior camera component.
`public Camera normalInterior`	This is the non VR interior camera component.
`public Transform focusPoint`	This is the transform/object which the exterior camera is programed to look at and rotate around.
Zoom	

`public bool zoomEnabled`	Specifies if the zoom functionality is active and enabled on the interior camera
`public float zoomSensitivity`	This is a measure of how fast the FOV is changed based on the mouse scroll wheel motion.
`public float maximumFOV`	This is the minimum/lowest amount the FOV can be set to during the zoom motion.
`public float orbitDistance`	This is the distance of the exterior camera transform from the focus point object. The camera can be rotated around the focus object in a circle with this radius.
`public float orbitHeight`	This is the distance above the focus object which the exterior camera maintains while orbiting it.
`public float maximumInteriorVolume`	This is a value that determines the volume level of things heard in the aircraft/helicopter cockpit. E.g. how loud will the engine be to someone sitting in the cockpit.
`public float azimuthSensitivity`	This is a measure of how responsive the exterior camera in "Free Mode" is to horizontal mouse movement.
`public float elevationSensitivity`	This is a measure of how responsive the exterior camera in "Free Mode" is to vertical mouse movement.
`public float radiusSensitivity`	This is a measure of how responsive the exterior camera rotate motion is to the mouse movement.
`public float maximumRadius`	This is the distance of the exterior camera transform from the focus point object. The camera can be rotated around the focus object in a circle with this radius.
`public float viewSensitivity`	This is a measure of how responsive the interior camera rotate motion is to the mouse movement.
`public float maximumViewAngle`	The maximum angle in the horizontal (left and right) direction which the interior camera rotation motion is clamped to.

INTERNAL FUNCTIONS

Initialize

Configures the bulb materials, sets the state and plots the flash/operation curve

AnalyseOrbitCamera

Configures the position of the exterior camera to rotate around the aircraft/helicopter in a fixed orbit and at a fixed height above the focus point.

AnalyseFreeCamera

Configures the position and rotation of the exterior camera which can rotate along the vertical and horizontal axes of the helicopter/aircraft based on the mouse inputs.

AnalyseInteriorCamera

Configures the rotation of the interior camera based on the supplied horizontal and vertical view inputs.

Compute (`double dt`)

Receives computation command from the controller script and evaluates the current intensity of the light bulb from the flash curve.

`double dt: This is the fixed timestep value from the controller. This value is sent from the controller to give the user max control over the integration/interpolation timestep.`

AnalyseCameraAngle

Calculates the position of the exterior camera relative to the center of the aircraft. This is used to control the sound to play at different points of the aircraft.

Call Functions

ActivateInteriorCamera

This function will switch the camera to the interior mode and go through the checklist of things to be done for the switch.

ActivateExteriorCamera

This function will switch the camera to the exterior mode and go through the checklist of things to be done for the switch.

ToggleCamera

This function helps you to switch between the Interior and Exterior camera modes.

SILANTRO PAYLOAD

`Oyedoyin.Common.SilantroPayload: UnityEngine.MonoBehaviour`

This component is used to mark objects that will have an effect on the weight and center of gravity of the aircraft e.g passengers, luggage or external attachments

PROPERTIES

Property:	Function:
`public PayloadType payloadType`	Specifies the type of payload this is. You can add more options to the enum and expand on them if you wish to.
`public CrewType crewType`	This option comes up when the payload type is set to crew. It determines what designation this crew member is.
`public WeightUnit m_weightUnit`	This specifies the base unit of the inputted payload weight. This will be used to convert the input weight value to kilograms.
`public float weight`	The mass of the payload in either kilograms or pounds.

SILANTRO PYLON

`Oyedoyin.Common.SilantroPylon: UnityEngine.MonoBehaviour`

This component is used to hold munition object and determine how they should be fired or ejected from the helicopter or aircraft.

PROPERTIES

Property:	Function:
`public PylonPosition pylonPosition`	Specifies if the pylon and munition connected to it are within or outside the aircraft/helicopter
`public LauncherType launcherType`	This describes the mode of operation of the launcher within this pylon
`LauncherType.Trapeze`	*This launcher will eject the missile attached to it first before launching.*
`LauncherType.Drop`	*This launcher will drop the missile from the bay before launch.*
`LauncherType.Tube`	*Here the launcher operates a rail system and the missile is fired off without adjusting its position.*
`public TrapezePosition trapezePosition`	Specifies the position of the trapeze launcher.
`public DropMode bombMode`	This is only relevant for pylons with bombs attached. This determines if the bombs should be released individually or in groups.
`public SilantroActuator pylonBay`	The actuator for the door connected to the pylon if it's an internal store.
`public float dropInterval`	This is the time in between sequential drops of the bomb in salvo mode.

CALL FUNCTIONS

START LAUNCH SEQUENCE

This function will start the checklist/process of launching or firing a missile connected to the pylon.

START DROP SEQUENCE

This function will start the checklist/process of dropping the attached bombs.

SILANTRO PLAYER

`Oyedoyin.Common.SilantroPlayer: UnityEngine.MonoBehaviour`

This contains a simple first person controller that can be used to activate the enter-exit functionality of the aircraft/helicopter. You can switch the movement and rotation logic to your custom controller.

PROPERTIES

Property:	**Function:**
`public Type m_type`	Specifies if the pylon and munition connected to it are within or outside the aircraft/helicopter
`Type.Default`	*The player will use the included simple first person control movement and rotation logic.*
`Type.Custom`	*You can specify the logic to drive the movement and rotation of the player object*
`public float m_walkSpeed`	This is the base movement speed of the simple player
`public float m_groundForce`	This is the maximum amount of force that will be applied to keep the player on the ground
`public float maxRayDistance`	This is the maximum horizontal distance from the player head transform to which an aircraft/helicopter can be detected. i.e. How far out from the head do you want to shoot the vehicle detection raycast.
Camera Control	
`public float azimuthSensitivity`	*This is a measure of how responsive the player camera is to horizontal mouse movement.*
`public float elevationSensitivity`	*This is a measure of how responsive the player camera to vertical mouse movement.*
`public float MinimumX`	*This the minimum vertical deflection to which the player camera rotation is clamped to.*

`public float MaximumX`	*This the maximum vertical deflection to which the player camera rotation is clamped to.*
`public Camera m_camera`	This is the camera component to be used for view by the player
`public Transform m_headPoint`	This is a reference position for the player head, this will be used to position the camera and the vehicle check raycast.

CALL FUNCTIONS

SILANTRO TOUCH

`Oyedoyin.Common.SilantroTouch: UnityEngine.MonoBehaviour`

This component will drive mobile joystick controllers which can be used to control the aircraft or helicopter.

PROPERTIES

Property:	Function:
`public Mode m_mode`	Specifies if the joystick returns to center point (0, 0) when it's released or stays where it was. A joystick to control the pitch and roll of the aircraft or helicopter will need to self-center while the throttle or collective stick will need to be static.
`Mode.SelfCentering`	*Returns back to the center (zero point) when it is released*
`Mode.Static`	*Stays where it is when it is released.*
`public AxisMode m_axis`	Specifies which of the joystick axes inputs are collected from.
`AxisMode.Both`	*This joystick will process inputs in both X and Y axis*
`AxisMode.XOnly`	*This joystick will only process inputs on the X-Axis*
`AxisMode.YOnly`	*This joystick will only process input on the Y-Axis*
`public InputMode m_xAxisMode`	Specifies if the input from the X-axis is taken normally or inverted.
`public InputMode m_yAxisMode`	Specifies if the input from the Y-axis is taken normally or inverted.
`[SerializeField] private RectTransform m_case`	
`[SerializeField] private RectTransform m_ball`	This is the component whose position is going to be measured and transformed into input variables.
`[SerializeField] private readonly float deadZone`	This is the threshold above which the input will be recognized i.e. the input has to be above this value to be considered valid.

`[SerializeField] private readonly float m_range`	This is the maximum amount the ball transform is allowed to move from the center point.
`public float m_centerSpeed`	This is the speed with which the self-centering joystick goes back to 0, 0

SILANTRO RADAR

`Oyedoyin.Common.SilantroRadar: UnityEngine.MonoBehaviour`

This component will track aircrafts, munition and any object in the scene with a transponder component attached to it. It also handles targeting and fire controls for the aircraft/helicopter weapons system.

PROPERTIES

Property:	Function:
`public ControlState m_advancedTracking`	Specifies if the radar can filter and help the aircraft fire munitions. i.e. does the radar have targeting and fire control functionality.
`public float m_range`	This is the maximum range in meters from the radar position to which an object can be tracked. i.e. objects beyond this distance will not be seen on the radar.
`public int m_maximumObjects`	This is the maximum number of objects that can be tracked and have their properties stored by the radar.
`public float pingRate`	This specifies how fast the radar sweeps to check for objects. A higher value will mean better tracking but come at a performance cost.
`public LayerMask m_collisionLayers`	This specifies the object layers the radar can check from. Object not included in these layers will be ignored by the radar sweep.
`public float size`	This specifies how big the radar object appears on the screen.
`public float Transparency`	Determines the transparency of the GUI radar draw object.
`public float objectScale`	Determines how big the tracked objects appear on the radar screen.
`public bool m_markTargets`	The object currently being tracked by the radar will be marked with the gizmo line component.

public *Texture* background	This is the texture that will be drawn on the screen as the radar background or base object.
public *Texture* compass	This is the texture that will be drawn on the screen as the radar compass or needle object.
public *Texture2D* selectedTargetTexture	This is the texture that will be drawn on the radar screen on the selected object.
public *Texture2D* lockedTargetTexture	This is the texture that will be drawn on the radar screen on the locked object.
public *Texture2D* TargetLockOnTexture	This is the texture that will be drawn on the camera screen for each track object in the radar range.
public *Texture2D* TargetLockedTexture	This is the texture that will be drawn on camera screen for the locked target.
public *Texture2D* m_aircraftTexture	This is the texture that will be drawn on the radar screen when the detected object is an aircraft.
public *Texture2D* m_missileTexture	This is the texture that will be drawn on the radar screen when the detected object is a munition.
public *Camera* currentCamera	This is the current camera component from the main camera controller (SilantroCamera) i.e. either the interior or exterior camera.
public *Camera* targetCamera	This camera component will be position to look at the current object being observed by the radar. You can render this into the cockpit and have proper view of the radar objects in the aircraft.
public *Camera* lockedTargetCamera	This camera component will be position to look at the locked object. You can render this into the cockpit and have proper view of the radar locked object in the aircraft.
public float cameraDistance	This is how far from the current target the camera is positioned.
public float cameraHeight	This specifies the height above the target object to position the target camera.

Call Functions

ROCKET MOTOR

`Oyedoyin.Common.SilantroComponents.RocketMotor: UnityEngine.MonoBehaviour`

This component is attached to the munition object to provide the needed thrust and propulsion for them to reach operational speeds.

PROPERTIES

Property:	Function:
`public BurnType burnType`	Specifies the burn curve to be used by the rocket motor. The burn curve is a plot of the mean thrust value vs the burn time.
`BurnType.Neutral`	*This motor will burn at a constant rate for the whole burn time.*
`BurnType.Regressive`	*This motor will burn high at first and then gradually get lower over time.*
`BurnType.Progressive`	*This motor will start out low and then increase as the burn time increases.*
`public float m_meanThrust`	This is the maximum thrust the motor can reach over its complete burn time.
`public float fireDuration`	Specifies the total time the rocket motor will burn for.
`public ParticleSystem exhaustSmoke`	This is the particle effect for the exhaust smoke from the rocket motor.
`public ParticleSystem exhaustFlame`	This is the particle effect for the exhaust flame from the rocket motor.
`public float maximumSmokeEmissionValue`	This is the maximum emission level the smoke particle will reach at maximum thrust level.
`public float maximumFlameEmissionValue`	This is the maximum emission level the flame effect will reach at maximum thrust level.
`public AudioClip motorSound`	This is the audio clip to be played when the rocket motor is active.

`public float maximumPitch`	Maximum pitch level of the rocket motor audio source.

SILANTRO EXPLOSION

`Oyedoyin.Common.SilantroExplosion: UnityEngine.MonoBehaviour`

This component is used to instantiate the correct prefab for an explosion, control the sound and light component.

PROPERTIES

Property:	Function:
`public float damage`	Specifies the amount of damage to be applied to the objects within the explosion range.
`public float explosionForce`	Specifies the amount of force to be applied to the objects within the explosion range.
`public float explosionRadius`	This is the maximum distance from the explosion center to start checking collision objects from.
`public float exposureTime`	This specifies the amount of time the light component from the explosion will be visible for.
`public float lightIntensity`	Specifies the maximum intensity for the explosion light component.

SILANTRO GUN

`Oyedoyin.Common.SilantroGun: UnityEngine.MonoBehaviour`

Normal gun component. Can be configured to fire raycast or rigidbody bullets.

PROPERTIES

Property:	Function:
`public BulletType bulletType`	Specifies the logic to use for the bullet system. Will it shoot raycast or actual rigidbody bullets?
`BulletType.Raycast`	*This gun will use raycast logic for the bullet action.*
`BulletType.Rigidbody`	*This gun will shoot the specified rigidbody bullet.*
`public Transform[] muzzles`	Specifies the transforms whose position, bullets will be shot from.
`public Transform barrel`	This is the barrel transform which should be a parent of the muzzle objects. It can be specified to rotate.
`public GameObject bulletCase`	This is the prefab for the bullet case that will be instantiated when the gun is fired.
`public Transform shellEjectPoint`	Specifies the position from which the gun bullet case will be instantiated from.
`public float rateOfFire`	Specifies how fast the gun fires in rounds per minute. This will be capped based on the assigned project fixed update time e.g if max fixed update = 0.02, the maximum rate your guns can fire is (1/0.02 = 50rps, 50 * 60 = 3000 rounds per minute.
`public float accuracy`	This is gun accuracy in percentage (0 - 100%).
`public float accuracyDrop`	How much accuracy in % does the gun lose each fire step
`public float accuracyRecover`	How much accuracy in % does the gun gain each time step (as it cools down)

`public float muzzleVelocity`	This is the velocity in m/s with which the bullet leaves the gun barrel.
`public float barrelLength`	Length of the gun barrel in meter, used to calculate the recoil force.
`public float gunWeight`	Mass of the gun and its components (minus the bullets) in kilograms
`public float damperStrength`	Percentage of the recoil force absorbed by the gun dampers.
`public int ammoCapacity`	Maximum amount of bullets the gun can carry.
`public bool unlimitedAmmo`	Specifies if the gun should have unlimited ammo.
`public AudioClip fireLoopSound`	Specifies the sound effect/audio clip to play as the gun loops through its fire function.
`public AudioClip fireEndSound`	Specifies the sound effect/audio clip to play when the gun ends its fire function.
`public float soundVolume`	Maximum volume of the gun sound effects.
`public float soundRange`	Maximum distance in meter from the gun where the sound can be heard from.
`public GameObject muzzleFlash`	Specifies the gun muzzle flash prefab/particle effect.
`public GameObject groundHit`	Specifies the hit effect/particle effect to be instantiated when the object hit by the bullet is tagged “Ground”
`public GameObject metalHit`	Specifies the hit effect/particle effect to be instantiated when the object hit by the bullet is tagged “Metal”
`public GameObject woodHit`	Specifies the hit effect/particle effect to be instantiated when the object hit by the bullet is tagged “Wood”

CALL FUNCTIONS

SILANTRO MUNITION

`Oyedoyin.Common.SilantroMunition: UnityEngine.MonoBehaviour`

This is basically a weapon component (bullet, rocket, missile or bomb) that can be fired at or dropped on a tracked object.

PROPERTIES

Property:	Function:
`public MunitionType munitionType`	Specifies the type or functionality of the munition.
`public AmmunitionForm ammunitionForm`	Describes the shape of the munition nose, this helps to determine what drag coefficient to use for the munition.
`public LayerMask m_collisionLayers`	This specifies the object layers the munition trigger can check from. Object not included in these layers will be ignored.
`public float munitionWeight`	For Rockets, Missiles or Bombs this is the mass of the munition in kilograms, for bullets this is the mass in grains.
`public float munitionDiameter`	This is the diameter of the munition in meters.
`public float maximumRange`	Maximum distance in meters the munition is allowed to travel without a defined target before it self-destructs.
`public float activationDistance`	This is the distance from the aircraft or helicopter beyond which the munition is armed i.e. the munition needs to be above this distance from the aircraft before it is armed.
`public float maximumSpeed`	This is the maximum speed of the munition in Mach
`public float ballisticVelocity`	This is the speed in m/s with which the munition (rocket or bullet) leaves it fire point
`public float detonationRange`	Minimum distance from the target below which the munition is detonated.
`public float destroyTime`	Maximum lifetime of the bullet once it's fired from the gun.

public *GameObject* explosionPrefab	Prefab with the explosion particle effects to be instantiated when the munition collides with the target or when it is detonated.
public float minimumLockDirection	Measure of how far from the target line of sight the missile is allowed to track from. Target direction directly facing the munition has a gain value of 1, while gain is 0 if target is 90 (or more) to the right or left.
public float navigationConstant	Proportional gain constant used by the navigation system i.e how much the missile turns based on the error value.
public float maximumTurnRate	Maximum rate in degrees per second by which the missile can turn/maneuver towards the target.

CALL FUNCTIONS

SILANTRO INPUT

`Oyedoyin.Common.SilantroInput: UnityEngine.MonoBehaviour`

This component will collect the input variables from various sources e.g. Keyboard, Joystick, VR or custom and process them into control variables for the flight computer. It also contains some control and command functions for the aircraft/helicopter.

PROPERTIES

Property:	Function:
`public Controller _controller`	Specifies the vehicle (aircraft or helicopter) which the input component is connected to.
`public float _pitchScale`	This value controls the pitch input curve which describes the relationship between the raw input value and the value the aircraft/helicopter receives. It can be used to amplify or deaden control inputs around the centre.
`public float _rollScale`	This value controls the pitch input curve which describes the relationship between the raw input value and the value the aircraft/helicopter receives. It can be used to amplify or deaden control inputs around the centre.
`public float _yawScale`	This value controls the pitch input curve which describes the relationship between the raw input value and the value the aircraft/helicopter receives. It can be used to amplify or deaden control inputs around the centre.
`public float _pitchDeadZone`	This is the threshold above which the input will be recognized i.e. the pitch input has to be above this value to be considered valid.
`public float _rollDeadZone`	This is the threshold above which the input will be recognized i.e. the roll input has to be above this value to be considered valid.
`public float _yawDeadZone`	This is the threshold above which the input will be recognized i.e. the yaw input has to be above this value to be considered valid.

`public float _pitchTrimDelta`	This is the step increase or decrease for the pitch trim input i.e. each trim step, this delta value will either be added or subtracted from the trim input.
`public float _rollTrimDelta`	This is the step increase or decrease for the roll trim input i.e. each trim step, this delta value will either be added or subtracted from the trim input.
`public float _yawTrimDelta`	This is the step increase or decrease for the yaw trim input i.e. each trim step, this delta value will either be added or subtracted from the trim input.

SILANTRO WHEEL

`Oyedoyin.Common.SilantroPiston: UnityEngine.MonoBehaviour`

This component controls the aircraft/helicopter brakes, proper positioning and rotation of the 3d wheel transforms.

PROPERTIES

Property:	Function:
`public BrakeState brakeState`	Specifies whether the aircraft/helicopter parking brake is engaged or not.
`public float brakeTorque`	Maximum brake torque applied to the wheels when the parking brake or brake pedal is engaged.
`public float brakeInput`	Brake pedal input in percentage (0 – 100%)
`public float maximumRudderSteer`	Maximum steer angle that can be commanded by the rudder pedals.
`public float maximumTillerSteer`	Maximum steer angle that can be commanded by the tiller lever.
`public float maximumSteerAngle`	This is the maximum steerable angle for the wheels, will be used to clamp the sum of the rudder and tiller steer angles.
`public bool pedalLinkageEngaged`	Determines if the rudder pedal steer is currently engaged.
`public AudioClip groundRoll`	This sound clip will be played on loop when the wheel is in contact with the ground and rolling fast enough.
`public AudioClip brakeEngage`	This sound clip will be played when the brake engage function is called.
`public AudioClip brakeRelease`	This sound clip will be played when the brake release function is called.
`public float maximumExteriorVolume`	Maximum volume for the wheel and brake sounds on the exterior camera listener.

`public float maximumInteriorVolume`	Maximum volume for the wheel and brake sounds on the interior camera listener.
`public WheelCollider leftBalanceWheel`	This wheel collider wheel be used as the left position beacon to apply the anti-roll force.
`public WheelCollider rightBalanceWheel`	This wheel collider wheel be used as the right position beacon to apply the anti-roll force.
`public float AntiRoll = 5000`	Maximum force that can be applied on the left and right wheels during a turn to keep the vehicle from flipping over.

CALL FUNCTIONS