uSim *1.32b*

Vehicle simulation framework for Unity 5

About uSim framework:

uSim framework is ment to provide a very flexible aprouch to simulating core vehicle behaviours for both gameing and serius simulations porpuses. This is achieved trough 10 main vehicle components.

- Airfoil
- Highlift device
- Suspension
- Wheel
- Engine
- Proppellers
- Custom airdrag object
- Gearbox
- Differential
- Buoyancy
- Compartments

This components can be combined to recreate different vehicles configurations. From normal airplanes to seaplanes, from street cars to amphibious vehicles.

uSim ultimate goal is to be able to cover all vehicle types configurations on the air, land, sea and even space, providing at the same time all the structure to run a simulator or game.

The idea is that the client then just needs to add gameplay and content to the intended game or simulation.

Because the framework is very flexible it takes some time to test and tweak vehicles when attempting to match real world data. For this reason as many templates as posible has been added and will keep being added to the library to be used as guide.

It is also in the plans of adding testing applications for the the vehicles such as wind tunnel simulation and car dyno to avoid excesive time loss when testing new vehicles. Ofcourse it's allways fun to take the vehicles for a ride.

Also for arcade like porpuses the framework can be used by adjusting some key parameters in the simulation. This different aproches are addressed in the documentation.

This document will cover the new Air module 1.3. until Land and Sea modules are also fully migrated and updated.

AIR MODULE 1.3

Document Contents

Air module

- **Creating a new aircraft.**
- **Setting up the 3d model.**
- **Setting up aerodynamic components.**
- **Setting up undercarriage components.**
- **Setting up engines/props and fuel components.**
- **Setting up general components.**
- **Flight model configuration (Vehicle editor window).**

Please note, by default all vehicles included in this package are located at Libraries/vehicles/ in the Usim assets folder. Is strongly recomended to keep this distribution for your vehicles.

Located at Configuration files folder, is included the inputsManager proyect configuration file for use with unity standard inputs.
To use it, replace InputsManager.asset file located in (your project name)\ProjectSettings folder.

Aircraft components list.

Description and requeriments by category :

Controllers

- **AircraftController:** Controls general aircraft characteristics.
- **AircraftSounds:** Controls wind sounds.
- **LandingGearAnimation:** Controls landing gear states. Requieres target Animator component.

Aerodynamics

- **CurvesManager:** Holds all airfoils coefficients curves data. The core of the aerodynamics behaviour.
- **Aerofoil:** Simulates lift and drag forces of an air foil. Requieres a target rigidbody to be assigned and a CurvesManager.
- **AerofoilSound:** Controls turbulent flow sound on an airfoil for sound FX. Requieres a soundSource.
- **HighliftDevice:** Modifes airfoil properties of an aerodynamic component to simulate wing shape shift. Such as ailerons, elevators, rudders, flaps, etc.
- **ControlAnimator:** Articulate the moving surfaces of the airfoils (the HighliftDevices

Transforms). Requieres an InputsManager.
- **AirdragObject:** Custom air drag component. (also integrated as airbrake)

Props

- **Prop:** Static aircraft's propeller simulation. Uses output and friction curves. Physics based connection to engine. Requieres an Engine component.

Common uSim components

Engines

- **EnginesManager:** Wraps up all engines in the vehicle. Controls engine selection and states.
- **Engine:** Physics based engine. Simulates both torque and friction (engine brake).
- **EngineSound:** Controls engine sound samples and mix. Requieres an Engine and and AudioSource per engine sound sample.
- **EngineStarter:** Energizes the engine it is attached to. Torque can be adjusted. Requieres a target Engine and an AudioSource (can be muted).

Inputs

- **InputsManager:** Main class for recieving and hold input data for components to read from.
- **VehiclesManager:** Example class to handle active vehicle data.
- **StandarInputs:** Example class to wrap up unity standard inputs and assign to vehicle inputs. Requieres VehiclesManager.
- **MouseJoystick:** Example class to use screen mouse joystick as input. Requieres VehiclesManager.
- **RewiredInput:*** Example class to support Rewired inputs. Requieres Rewired Asset libraries installed.

* Rewired is a 3rd party asset and is not included in this package. You can buy it from the author at the Unity Asset Store. Usim just includes code soultion for handling inputs and does not include any Rewired script or related asset. Installing this extension with out having Rewired previusly installed will cause compile error.

Undercarriage

- **VehicleWheel:** uSim physics based wheel collider. Requieres AudioSource for slip sound FX.
- **VehicleSuspension:** Physics based suspension arm. This solution requieres to have a VehicleWheel attached.
- **RollSound:** Controls roll sound of the wheel. Requieres roll AudioSource.

Systems

- **FuelManager:** Wraps up vehicle fuel compartments information and handles fuel tank selection.
- **FuelCompartment:** Simulates fuel behaviour and weight.

General / globals

- **AtmosphereGlobals:** Holds air density chart and usage for the airfoil components in scene. Requieres to be in scene.
- **UsimVehicle:** Class to hold aircraft entity data such as name, description, panel reference and other general data. It is requiered for all uSim vehicles.

Creating a new aircraft

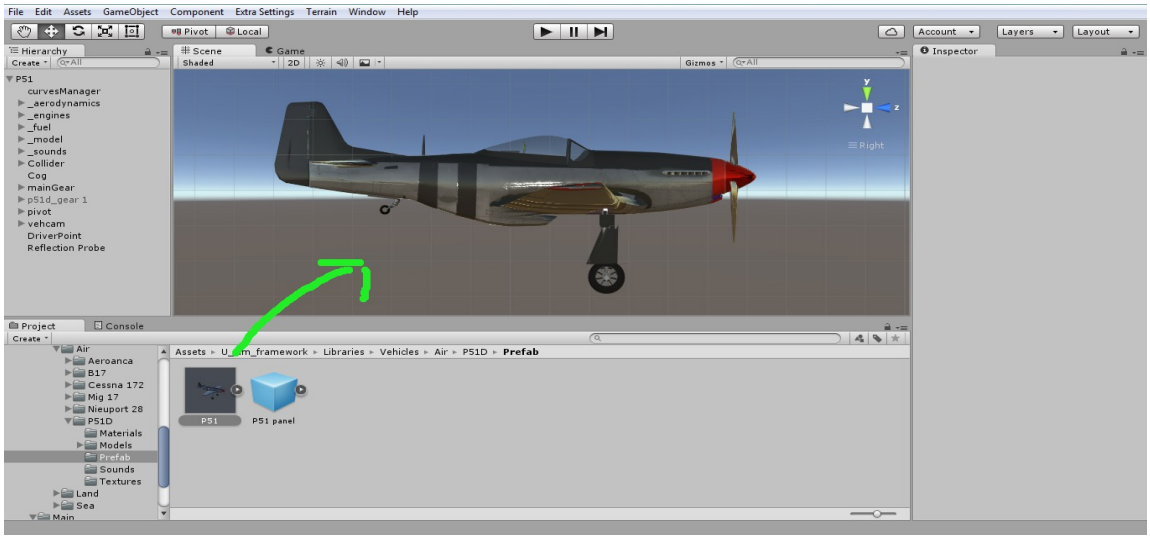
To create a new aircraft go to Window → Usim → New vehicle. in the editor menu.
This action will create a new folder in the vehicles library folder in the proyect. The new folder will also contain sub-folders in order to keep the assets organized. The folders are...

Materials
Models
Prefab
Sounds
Textures

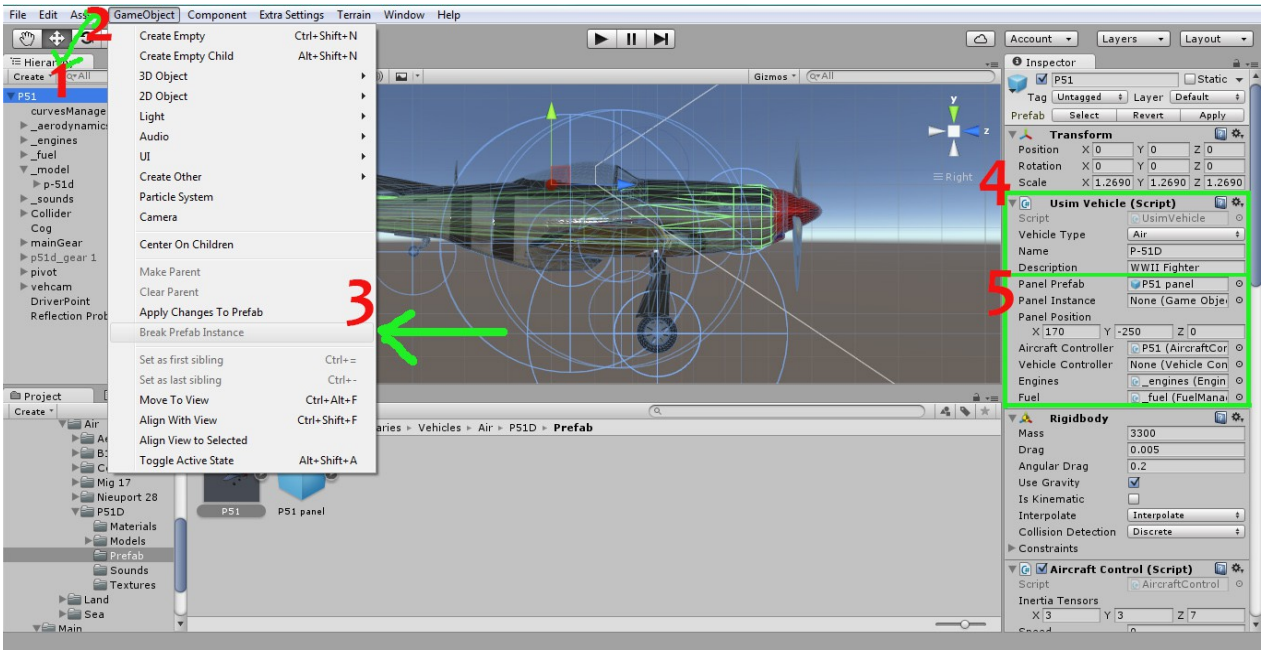
Starting an aircraft from scratch can be a time consuming task and complex to setup all needed components. But uSim is designed to use any provided template as starting point and just adjust components positions and parameters.
The key factors to having a clean flight model is to be precise in the positioning of the components and making sure to keep aircraft symetry.
It is required for the used to know how to manipulate and use unity 3d enviroment and inspector.

Important: This document covers the setup of the aircraft model, however another document is included that covers tips and sugested workflow on achieving the desired flight model behaviour.

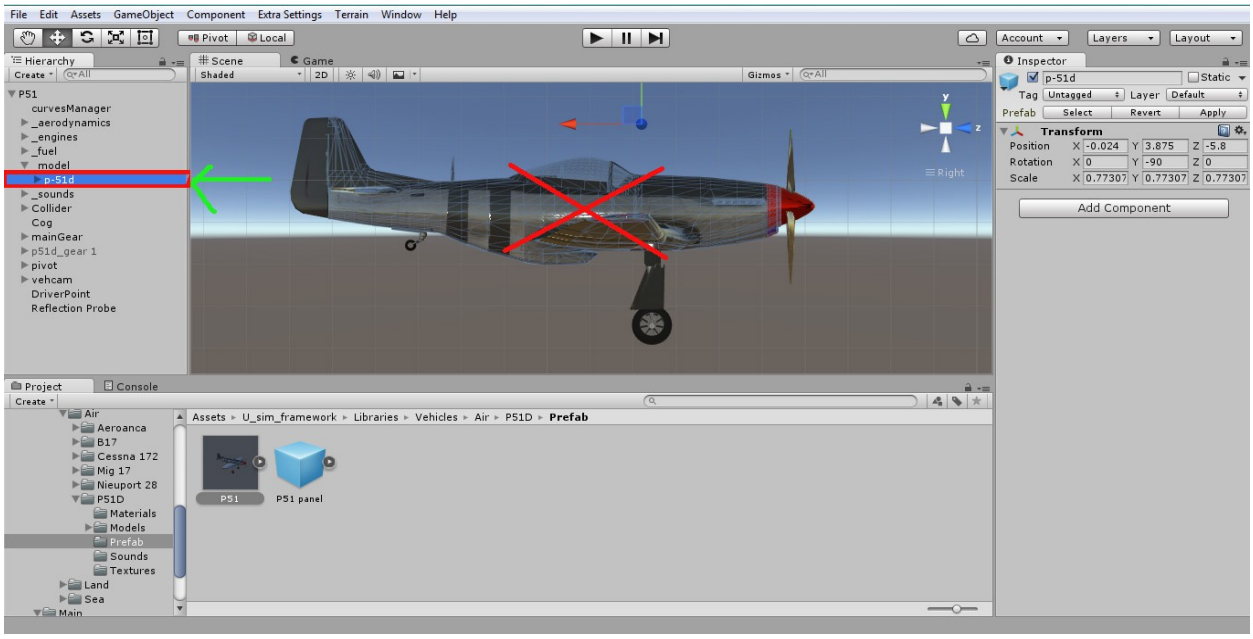
To start with, select form the vehicles librarie the aircraft wich configuration aprouches more to the aircraft you want to simulate. Drop into scene the prefab gameObject located in the prefab folder.
To make sure the aircraft is aligned to the world grid for masure porpuses, select the root object and set the transform position to 0,0,0 and the rotation to 0,0,0 and the scale to 1,1,1



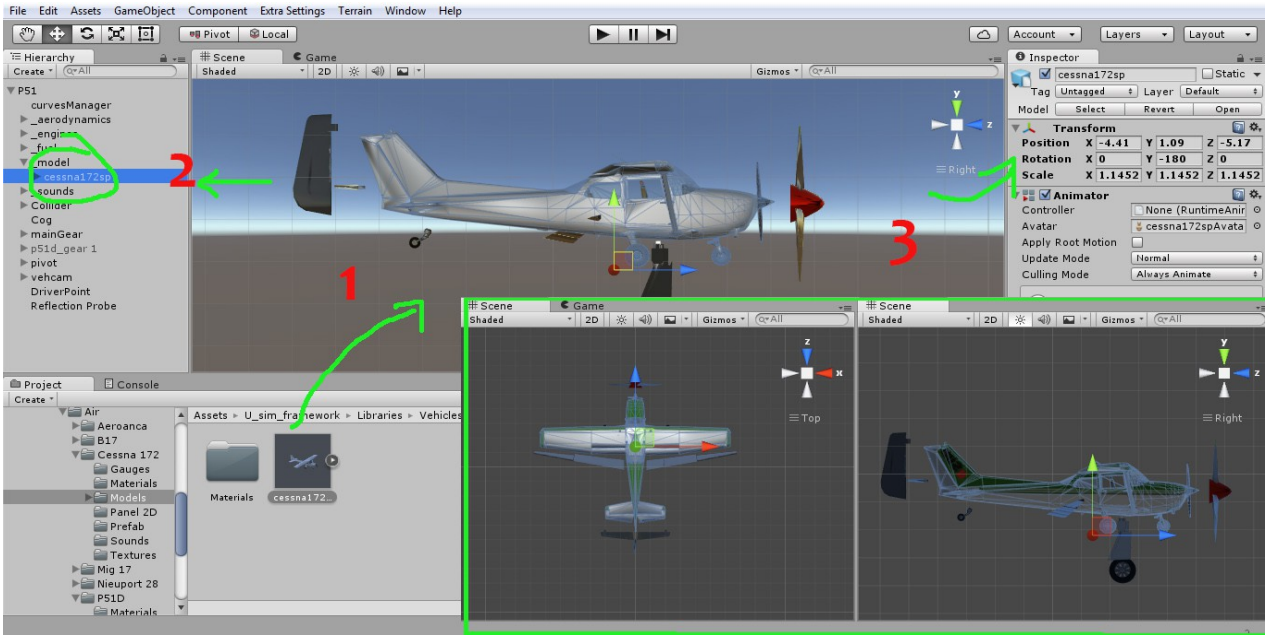
Next, to make sure you don't replace existing aircraft prefabs, brake the instance connection. To do so, select the aircraft from hierarchy tab, go to GameObject menu tab and select “brake prefab instance”. The gameObject name should be red (or black). Now is safe to modify it.
On the inspector tab, you can edit the uSim vehicle information on the UsimVehicle class.
Assign the 2D prefab to use. You can use any of the panel examples. Drag and drop in the inspector.
Panel position allows you to set the panel offset on screen.
IsSeaPlane can be turned on for the porpuse of identifying if the aircraft has buoyancy capability. This feature is use by the demo scene to tell if the aircraft can be spawned on water.



Next, delete from scene the aircraft 3d model so we can then use our own 3d model. Find _model object in hierarchy and expand to see the 3d model. Select and delete.

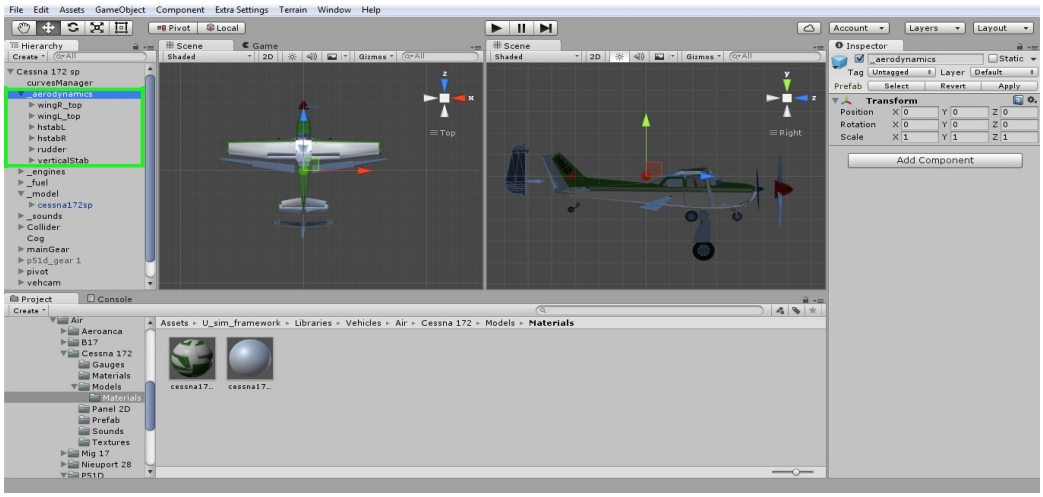


Now throw into scene or hierarchy your 3d model. Make sure to make it child of _model gameObject by draging it on top of _model.
IMPORTANT, here is the right time to properly set the aircraft dimensions. Use the grid to masure. Each unity grid unit is 1 meter. If you want to be more precise, create a cube object and scale it to the dimension you want in meters and use it to masure the aircraft size.
Also make sure the model is aligned in both top and side views.
Note that 3d model pivots can be wrong and just seting x component of transform position can not allways work. Make sure that it is visualy aligned.
Also when manipulating Transform objects in scene check allways that the handlers are set to Pivot and not center since we care for the position of the pivots most of the times.

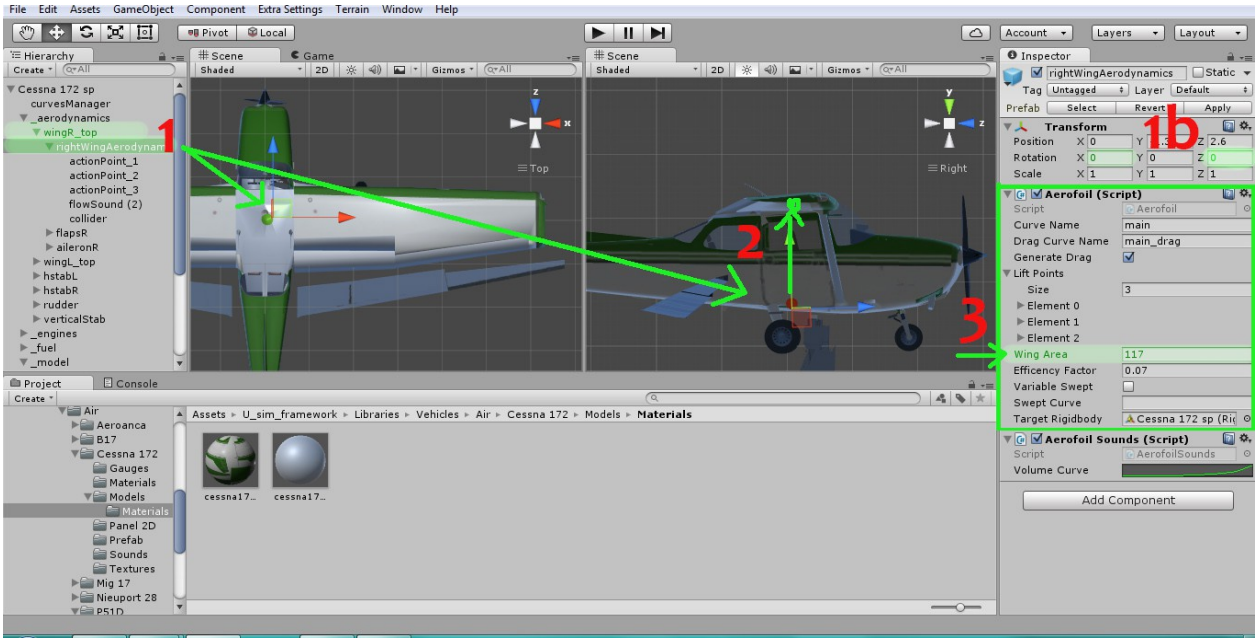


Seting up the aerodynamics.

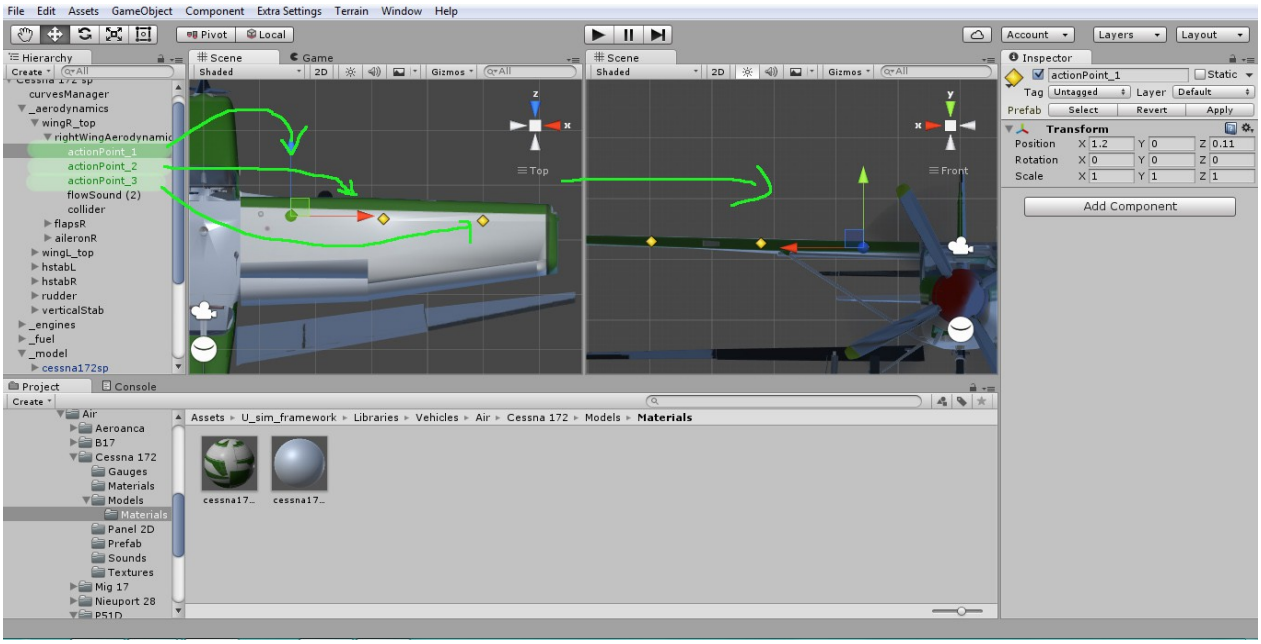
Next, expand the object called "_aerodynamics".
This object contains all the "wings" of the aircraft and their corresponding highlift devices.
Most commonly this are main wings, horizontal stabilizators and vertical stabilizators.



Each child object is the aerodynamic component root object. It is the point at where the airfoil is attached. Move this object to where the airfoil root should be. Make sure it is aligned in all axes. Once set, look for the other side partner if it has. And copy Transform component values. But then change the sign of the x component of the Transform position. This way you ensure Symetry.
Same happen with diedral angle. Make sure they have the same inclination angle.
On the inspector tab, put the name of the curve to be used for both lift and drag.
Also, set the surface area.
The efficiency factor is a number that affects overall L/D relationship by multiplying the generated drag.
Also set the target rigidbody. Generally this will point to the root object wich has the rigidbody component attached. But an alternative is to have the wing set as a rigidbody attached to the aircraft via joints and apply the foild force directly to the wing instead. The result is similar but it can be usefull to detach a wing and have the lost of aerodynamic forces along to simulate damage or wing ripping.
Also a wing can be simulated with various aerodynamic components if the wing has very diferent shapes along the length of the wing. Most comonly to optimize one would use just one with an aproximated average lift and drag curves. But using more than one aerofoil component does add realism to the overall wing behaviour.



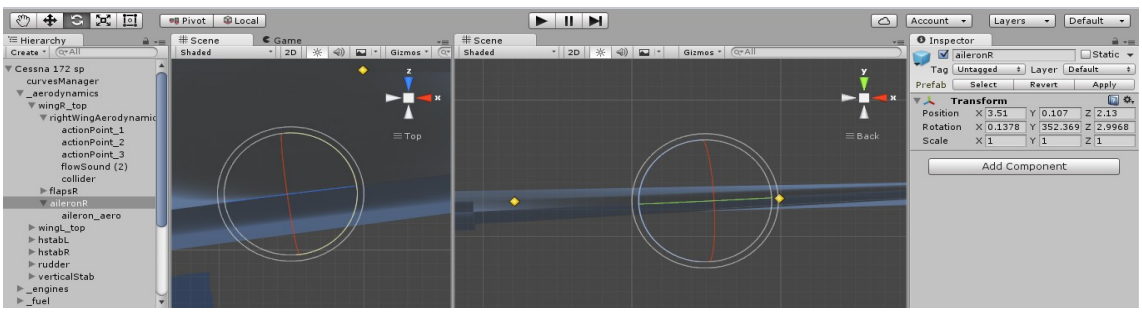
Each airfoil contains a sub class called LiftPoint. This are Transform references that are used to point the location at where a data iteraction is done. In other words, the points where the wing calculates and apply the forces.
Usually the points are set along the wing at the lift point wich tends to be the most thick part of the wing when seen from a side.
On the sideways position it is recomended to use 2 or 3 depending on the highliftDevices the wing has. Each liftPoint can be modified by a HighliftDevice object. So for example if the wing has a flap at the root and an aileron on the rest of the wing, just 2 points could be used.
Expand each lift point class to see the details. If the point is affected by a highlift device, toggle "variable shape" and assign a valid HighliftDevice component from the hierarchy. Usually you would have the shape modifier child of the airfoil.



Seting up Highlift Devices

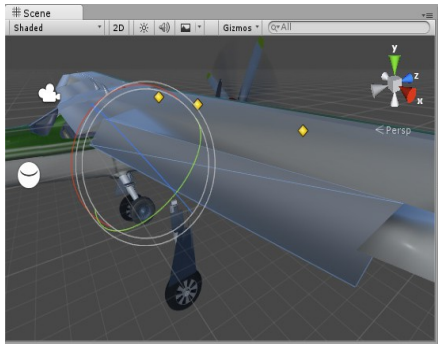
In the example we set the right aileron of the right wing. Select the aileron Transform and place it to where the aileron should pivot. Make sure the axis are aligned with the aileron like in the picture. Child of the pivot object you would place the HighliftDevice. This is because, for it to work, it needs to be deflected (animated). This is done by the ControlAnimator class.
As you may notice by now the patern is simple.

Wing/liftPoint → Aileron
(Airfoil) → (HighliftDevice)



Finally delete the old 3d model and move your aileron 3d model as child of the pivot point. You can now rotate the aileron along the X axis to see it in action to check if it is properly setup.

Repeat the same process in all the airfoils in aerodynamics.

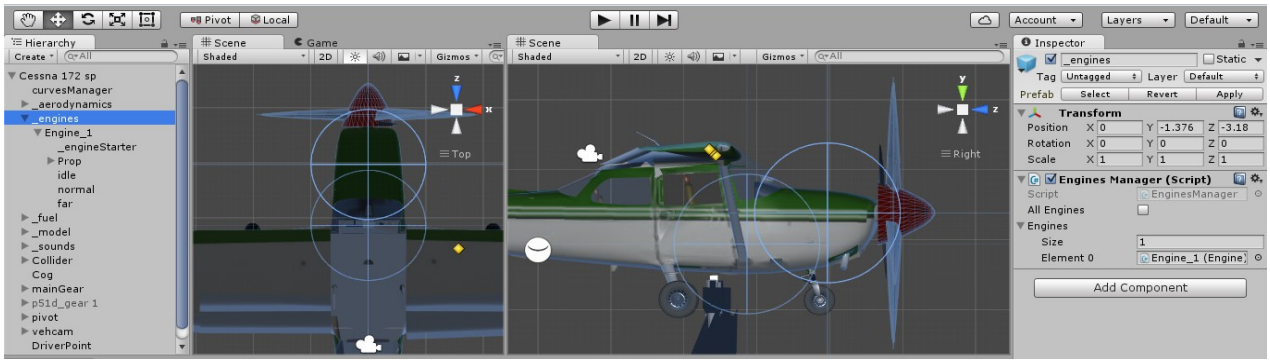


Setting up the curves manager references.

Before moving on, for the aerodynamics to work, a curve manager is needed. In the template is all allready setup but we need to change the coefficient curves to simulate our aircraft. You can find it in the aircraft hierarchy as “curvesManager”.
Again, this is adressed in more detail in the adjunted documentation “aerodynamics guide and tips”.
Generally speaking what we do here is setup diferent curves by it value and name so it can be accesed by the airfoils. If you remember you set the name of the curve in the airfoil inspector. This names should correspond exactly. You can copy and paste the value from one to other.
The coefficients are evaluated in function of angle of attack of the wing while flying. For now we will leave it as it is and will tweek the parameters later. The only important thing in this step is to make sure the names pointers are right.
We can now finally forget for now about the aerodynamics and move on to the engine, prop and fuel components wich simulate the tractor force of our vehicle.

Setting up the Engines and Engines Manager

Find the _engines object in hierarcy, it should have an EnginesManager component attached. Select it and in the inspector tab set the size of the engines and set the engine reference. The example template is allready setup for one engine, but in case you need to add more engines, set the size and reference for the new added engine.

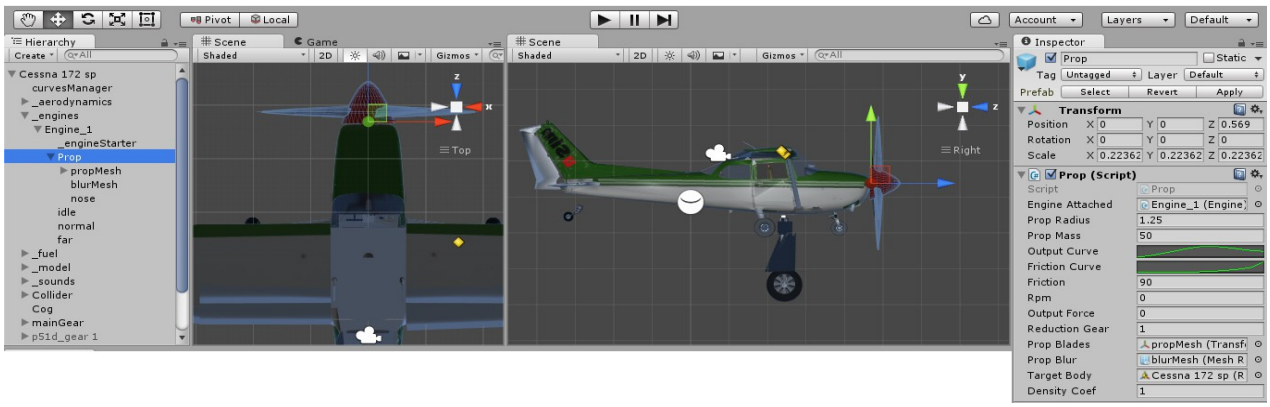


For any powered aircraft in uSim, two main components are needed. An Engine, and a prop. A prop is also used in jet engines simulating fan. Optionally more RPM can be developed by jet engines but

it's not really needed. Because of how the props are currently simulated it only adds static thrust. Means that it doesn't take into consideration the flow of the air trough the blades.
Dynamic prop simulation is planed for future versions as a separated optional component, since aircraft's prop is very critical in an aircraft flight model and requiere more complex applications.

The engine component has serval parameters in the inspector, but it is recomended to handle this parameters from the vehicle editor instead.
At this point all we need to know is that it has the right amount of engines and props and that are properly positioned and aligned.

For each Engine we have a Prop component attached. The prop's visual models need to be setup. To do this, expand the prop object and locate propMesh and blurMesh. PropMesh contains mesh rederers of the propeller blades at low speed. (solid). While blurMesh is a plane with a transparent blured prop texture set to it. This are interchanged by the prop component based in RPM.
For this step just make sure that propMesh and blurMesh references are properly set and that target rigidbody is set. As the rest of the components we will fine tweek them via the vehicle editor.



Tip: the cone or center of the prop can be detached from the blades so it is also seen when spining fast.

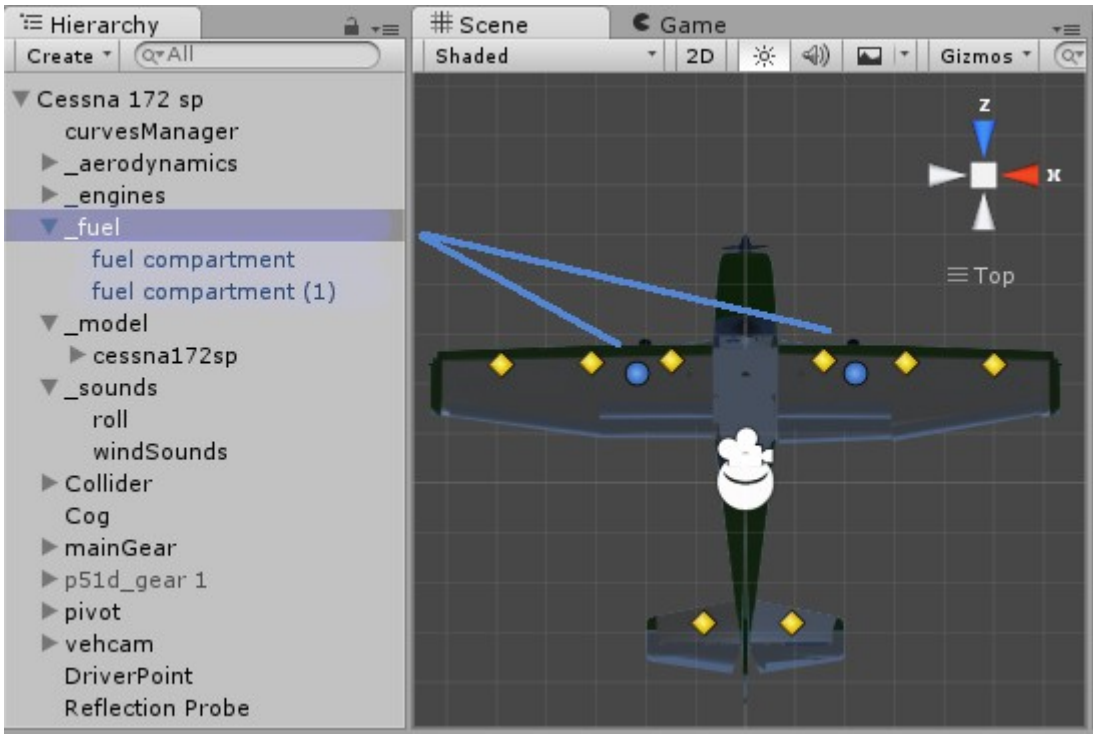
Engine sound

For the engine sound an EngineSound component has to be attached to the engine object. The template allready has it setup.
This component needs to have and Idle, normal and far sound samples. Each sample is stored in an AudioSource in the engine object hierarchy. EngineSound has to reference this to handle the volume and pitch.
Idle has as parameters the RPM reference of the sample. Rpm at wich volume starts fading out and rpm at wich sound ends fading out.
For normal is the same but since it comes after idle it has an rpm value at wich sound starts fading in and rpm at wich sound fully fade in instead.
Far sample is just a very far sound that is played in the background and when far. Is recomended to be used with low volumes.
Finally the reference to the engine component and pitch factor.
Pitch factor is an over all pitch factor. Usually this is tweeked in conjuntion with rpm samples. It is sugested to use values between 0.3 and 0.7. But depends greatly on the sound sample.
Also make sure the AudioSource parameters feets your requeriments.

Setting up fuel feed.

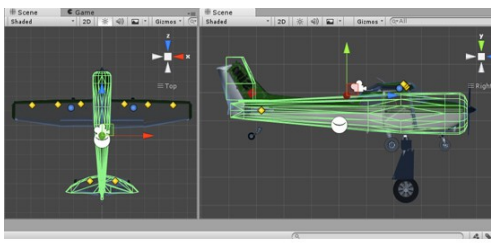
Fuel manager and fuel compartments

In the hierarchy, find the object called _fuel. It should have the FuelManager component attached. Like the engine manager, set the size and references to the fuel tanks.
Fuel tanks are child objects with the component FuelCompartment attached to them. Each fuel tank has a few parameters that we will tweek later trough the vehicle editor, at this point check for target rigidbody and proper location of the tanks. Take in mind that weight is added to the aircraft depending on the amount of fuel and fuel unit weight.



Setting up colliders.

In the hierarchy find the object Collider. Child to collider goes all the collider geometry. This meshes are used to compute physics. It is recomended to use as few as posible and very very low polly. Use of unity primitives shapes is good.
If you want to use a mesh collider, make sure you set it to be convex. Also remember that the more complex the mesh the more performance impact it has.



Setting up undercarriage

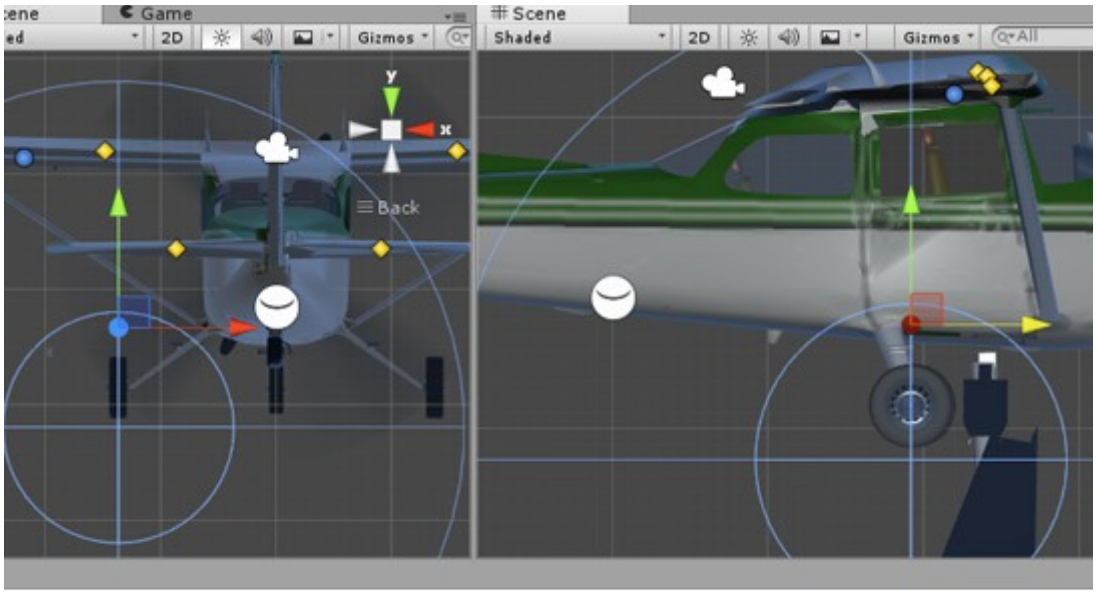
Almost done, now we need to set up the undercarriage. Meaning the wheels and suspensions as well as landing gear animation in case it is retractable.
Find the object mainGear in the hierarchy. If it has retractable gear it should have a LandingGearAnimation attached. In the example we don't need it so we remove it.

The gear animation is very simple solution and it will be improved. Basically you would have your gear 3d model animated normally and imported as legacy.
Drop the model and set it up in the main gear object so it looks in the right position. Then asign the Animation component to the LandingGearAnimation animation variable.

The static version of the landing gear (when on fully extended state) is the one that is physically simulated and articulated. If you want to use a diferent aprouch that animates the static gear structure refer to version 1.2

suspension setup

Select one of the suspension objects in te main gear hierarchy. It should have a VehicleSuspension component attached to it.
First to have things cleaner, remove all mesh from the template aircraft. Then, move it to the location where it would be the anchor of the suspension and then move it sideways so it is in top of the wheel. Like in the picture.

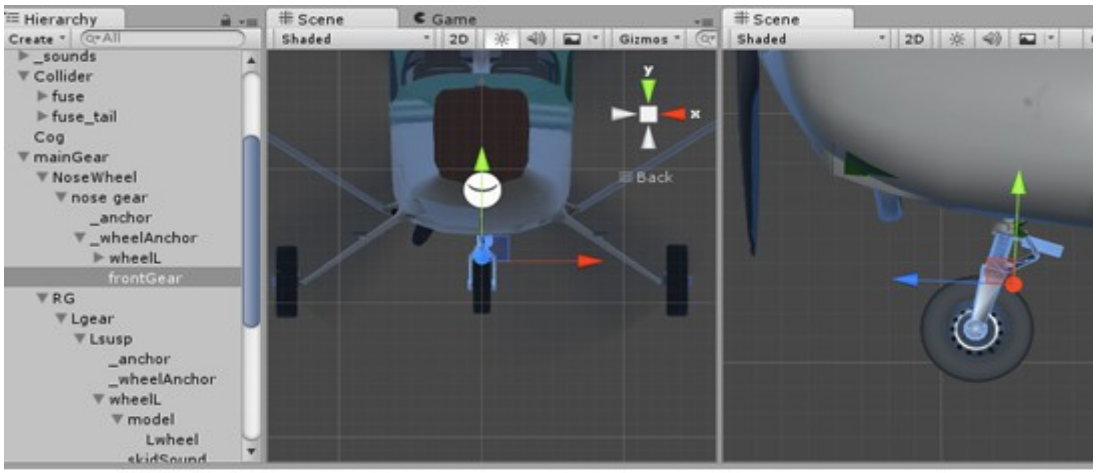


This may look wierd but since suspension components work vertically to the wheel at least in most cases, we place it aligned with the wheel. In real life cessna 172 case, that upward force is produced by the force of the landing gear arm when it bends like an elastic, but uSim suspension simulates clasical spring/shock model. As you can see the force is calculated diferently but the result is the same.

Next, set the Wheel object position to match as much as posible the wheel center. If the model if properly setup the wheel model pivot whould be at the very center. Select the wheel model and make it child of the object model in the wheel object hierarchy.
Once more there are serval parameters in both the suspension and wheel components. But we will tweek them via the vehicle editor.

Do the same with the other side gear.

Now the nose wheel is a bit diferent just because it has also a shock that moves along. So what we do is make it child of the wheelAnchor object. This way it will move along the suspension axis. For this case beause the suspension works linear, we align the suspension with the model arm.

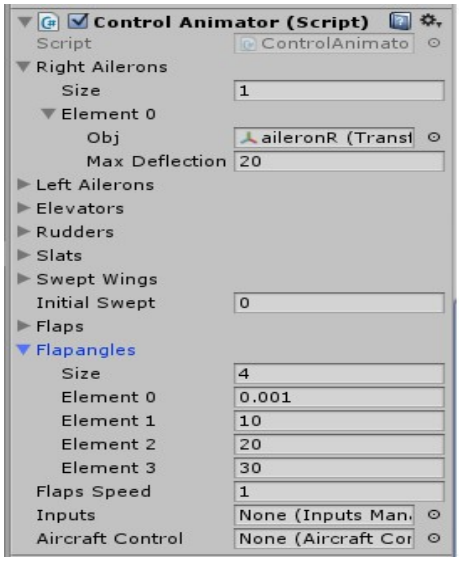


Setting up the animator controller.

One final thing before we have the aircraft ready, we have to now plug the ailerons, elevators, flaps and other movable parts of the aircraft. Mainly the highliftDevices we had setup.
Select the root object and find in the inspector a component called ControlAnimator.
You will see 8 expandable items. Each one contains an array of the transforms that will be articulated and the max deflection in deg. Note that this are preconfigured behaviours and each one has it own function and therefore the names.

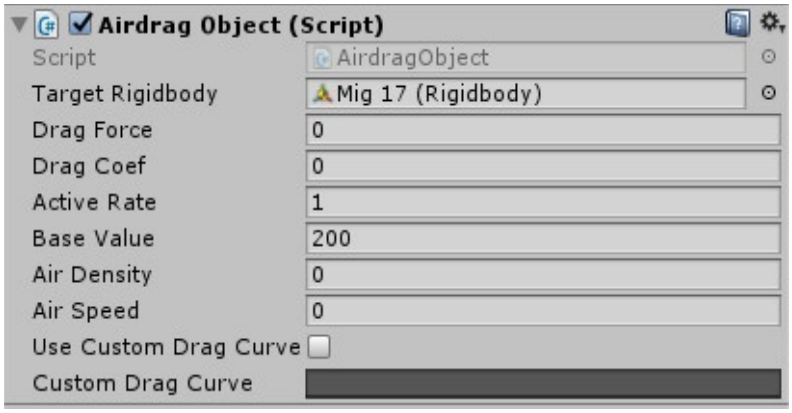
Right ailerons
Left ailerons
Elevators
Rudders
Airbrakes
Slats
Swept wings
Flaps
Flaps angles

As you can see the names are in plural.
So what you do here is as we did in the engines manager, set the size and assign the references and angles.
The only item that is diferent is flaps angles.
This is a list or array of the flaps angle presets.
10 deg, 20 deg, 30deg, etc.. each one is a flap position.
NOTE: beware that flap position 0 (wich means the first) has to be 0 deg so you don't have a default active flap.
Flaps deployment speed can be changed too.



Control Animator also includes interaction with airbrakes. Internally it alters the activation rate of the AirdragObject attached to the airbrakes.

AirdragObject.



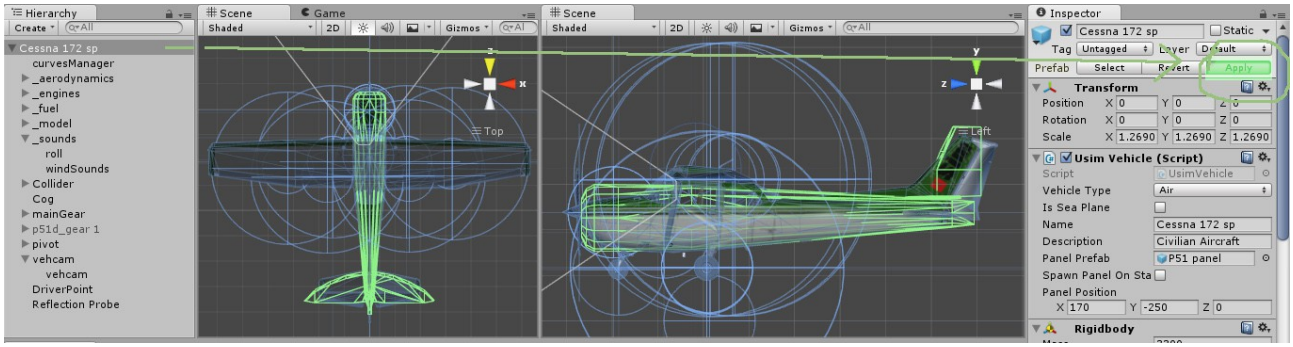
This object adds custom air drag based on airspeed. From here adjust the base value acordly.
If “use custom drag curve” is not check then drag coef of 1 is used.
Else you can use a custom curve where the Y axis is drag coefficient and the X axis is airSpeed / 10.

ActiveRate is the amount of “Visibility” of that drag part.
It is to be set externaly or left to a fixed value.
Landing gear animation script handles this value acordly if it is found atached to the static gear gameobject. This way extending the landing gear will gradually increase ActiveRate from 0 to 1 based on animation normalized time.

Camera stuff.

For the cockpit camera position we use as reference the object called pivot. This is the camera pivot point. Set it position to where the pilot head would be.

Finally, save your prefab to the prefab folder of your model. Do so by draging the aircraft into the folder view. After doing that you can allways use Apply button.



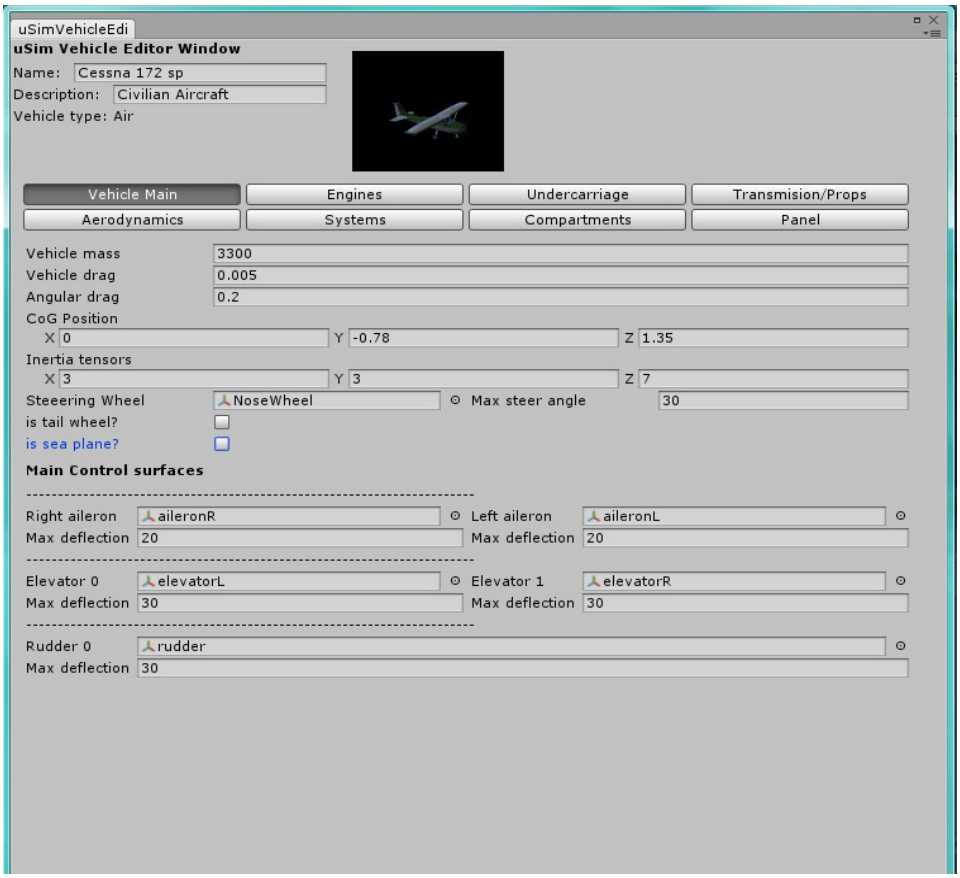
But mainly we will be using the vehicle editor view.

This concludes all the model setup. Now we have it suitable for tweeking via the vehicle editor.

Vehicle editor window

The vehicle editor window provides fast editing by gathering all editable values present in the aircraft hierarchy.
To open the vehicle editor window select the aircraft in the scene and go to Window → Usim → Edit vehicle

1 _Vehicle main

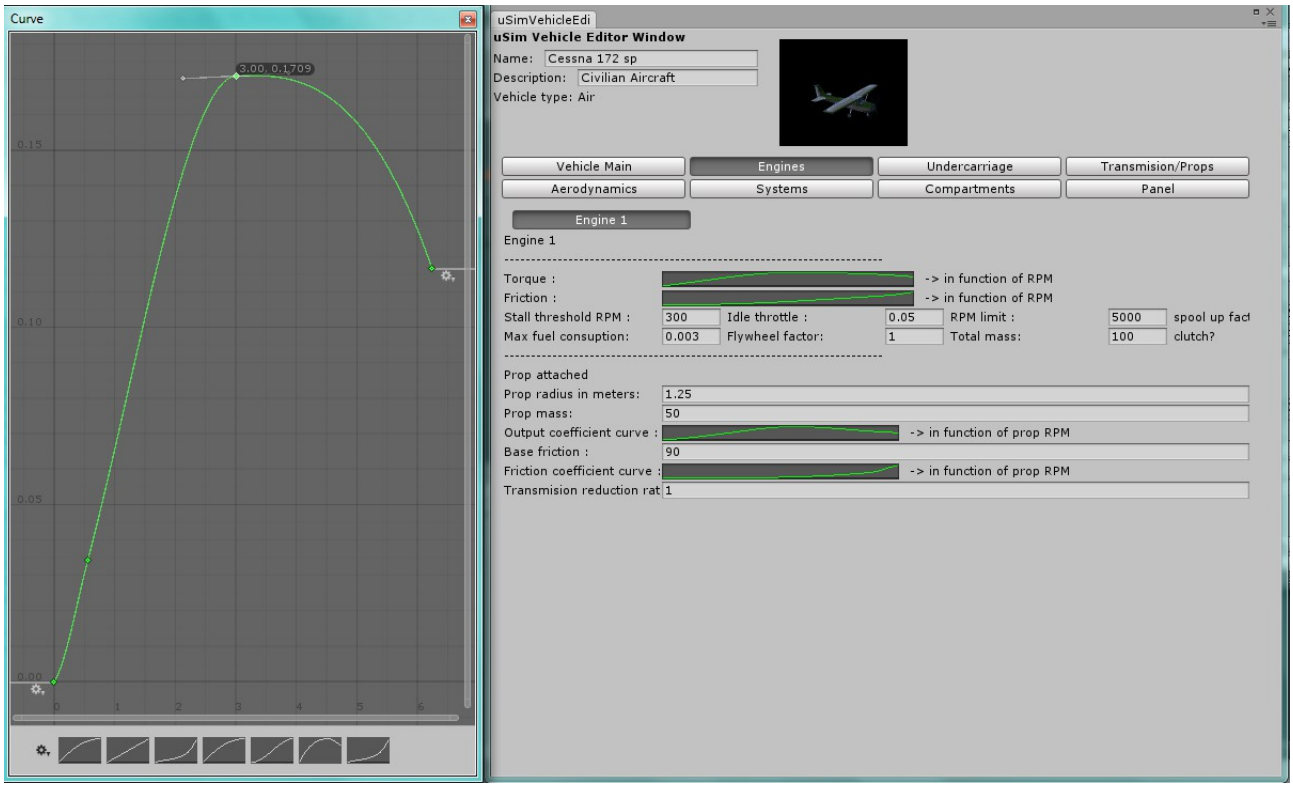


The first tab “Vehicle Main” refers to the relevant components attached to the root aricraft object. The first part refers to the unity's rigidbody configurations. Such as:

Vehicle mass : the “empty” weight in kg.
Vehicle drag: Omnidirectional static drag coefficient.
Angular drag: rotational drag coefficient.
Cog position: the local coordenades of the center of gravity point.
Inerta tensors: Not sure about the units here, but acording to unity it represents the amount of inertia along the corresponding axes. Inertia is how much “time” it takes the body to accelerate. It has to be with the shape of the body. In a box you would have the same amount of inertia in all axes. But in a cilinder the bigger inertia is along the longitudinal one (Z).
Steering wheel: points to the transform that will be rotated by the steering controller input.
Max steering angle: the max anlge the wheel can rotate in both ways. 45 = (-45 to 45)
Is tail wheel?: check this to have the steering inverted to control a tail wheel.
Is sea plane?: optional for identification.

The second part has to do with the control animator but it should be all setup by now. In case you need changing deflection angles you can do so from here.

2 _Engines



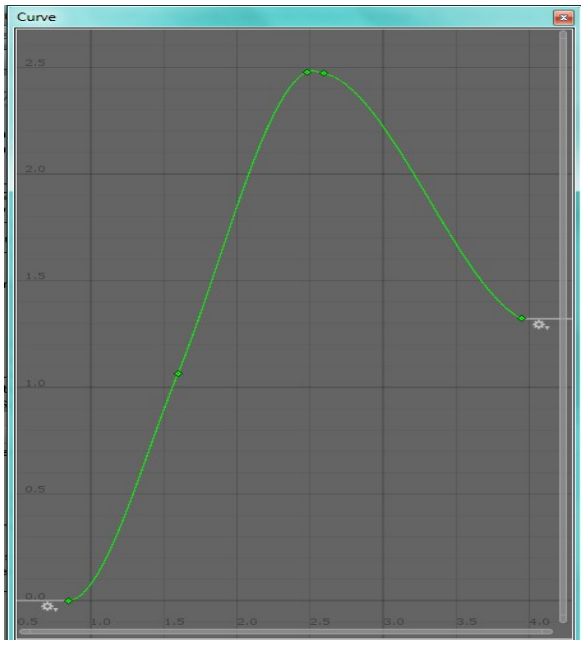
The second tab refers to all engines. You will see that for each engine found theres a tab.
At the moment when dealing with multiple engines it is better to use the aircraft in scene as we set it up before. And select all engines components at the same time and edit them all together. The vehicle editor is still mising this feature, but unity allready provides it that way. In our example we can use the editor window because it has just one engine.
Torque is curve that has rather a value of force * 1000. You can look at this as Hp * 1000 where 0,17 would be 170 hp. The engine output force is calculated internally so the curve factor behaves more like an hp value. This means the force the engine can produce to spin up itself.
In an empty space as in real life with out any friction the engine would spool up forever. But because the engine is made of physical components, it has gravitiy, and it has friction with other components of diferent densities.. the engine simulation also has a friction curve that also increases with the RPMs. In this model the engine would spool up towards the curve cross between torque and friction. Some friction comes from the prop attached.
Stall threshold is the RPM value at wich the engine sets the state to “stall”. This just cuts the RPM down to 0. An alternative is to set the torque curve in a way that it need some throttle to stay on. In real life this is actually the case. Because torque is represented as maxTorque * throttle input, if throttle is 0 hp will be 0 too, because it has no fuel.
Here is where Idle throtte parameter comes in. Again as in real life the engine allways need some fuel even to generate enough force overcome friction and stay at idle. In fact that's why the engine needs more fuel to start and keep at idle when it's cold, due to the increase on friction in the engine. Once the oil is hot and the engine has run a a while the friction reduces and therefore the engine needs less fuel to keep at idle. This complex relations are the next step in the engine simulation development. Hopefully some geometry based engine simulation and carburation simulation. Optional to add more realism to the engine component.

You can also limit the RPMs.

Spool up factor is how fast it reaches the target RPMs calculated by the engine. Simulates the inertia of the engine.
Max fuel consuption, is the rate of fuel consuption at max throttle as in units per second. This means maxConsuption * throttleInput.
Flywheel is a multiplyer of output force.
Total mass, the weight of all attached engine components. In other words, how much does the moving parts of the engine weight in kg.
Clutch toogle on if the engine uses the clutch input. The clutch set's the engine output and outside friction changes from current state to 0 based in clutch input. This is in place for the Land module and most likely not to be used in aircrafts unless you want to detach the prop from the engine via clutch input. Note that this will remove the prop friction making the engine to rev up.

Now comes the attached prop or fan.
Prop radius, this is to calculate the prop area, it should be the radius in mts of the circurference formed by the prop spining.
Prop mass in kg.
Now the output force curve is as the rest in function of RPM but this time the RPM of the prop.
Pops are just wings, but instead of being fixed, they rotate creating the forward speed they need to generate lift. In real life the prop section when spining can't go faster than the speed of sound because it loses lift due to sound barrier effects. In most aircrafts this happens at 2.500 rpm of the prop. It varies depending on the radius ofcourse, but for simplicity instead of doing all this calculations, we set the “lift coefficient” of the prop based on the prop RPM. If you know at wich speed the prop should start loosing efficiency you can make the curve drop at target RPM. Like this:

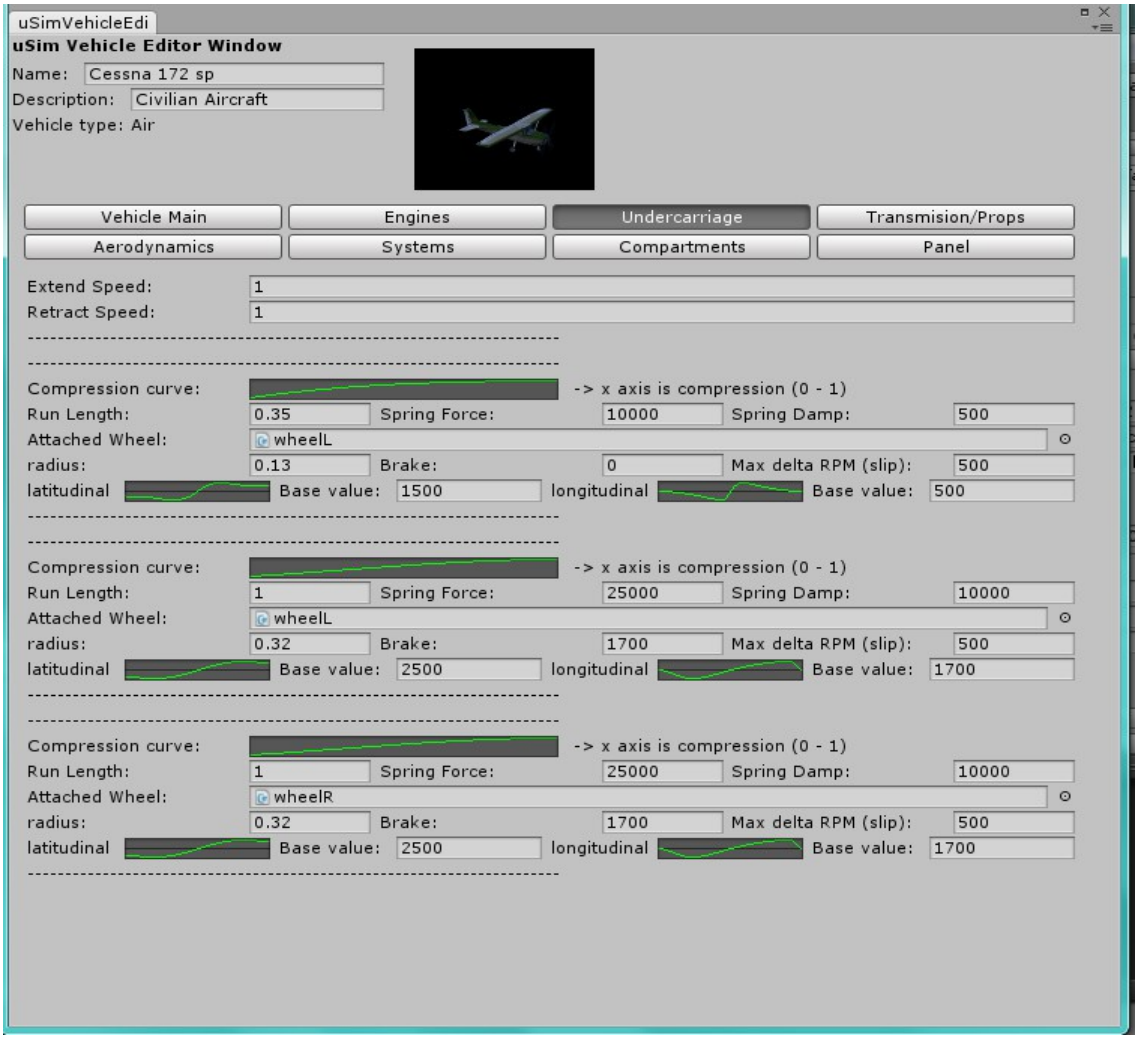
Note in the cessna we make that at 2.500 rpm the prop tip starts to loose efficiency. We clamp the low value to simulate that the inner parts of the prop are still under the speed of sound.



Friction is the same as the engine friction. But we specify the base friction value. So it's baseVal * friction coef.
Transmision reduction is the rate that the engine RPM is reduced into the prop. Some engine's may develop the torque at higher RPMs so for the prop not to go over speed of sound it needs to be reduced. This is what it's for.

3_Undercarriage

This is the last critical tweeking part. And perhaps the most time consuming.
In this tab you can tweek the suspension and wheel collider friction paramters.



In case the aircraft has retractil gear, you can adjust the animations speeds.
Compression: This is the value that represents how much the suspension is been compressed. The closer the wheel is to the suspension attach point the larger the compresion. Compression goes from 0 to 1.
So in the Compression factor curve you will set a parameter that would go from 0 to 1 generally in function of the compression rate.
So what does this compression factor? It multiply the Spring force. So it is springForce * comrpession coeff.
Run lenght is how much the suspension can extend. It is used to calculate the compression rate.
Spring damp is a force opposing the travel speed of the suspension. Adds biscosicty to it.
Attached wheel points to the wheel attached.
Radius of the wheel in mts.
Brake force, is the force that the brake has to oppose the wheel rolling force. If you want the brake to be able to block the tire, brake force should be greater than the longitudinal wheel force.
Max delta rpm is the amount of difference in rpm between a free spining wheel and the real rpm. This makes more or less sesitive the brakeing.

Last the latitudinal and longitudinal forces setup.
Base values are multiplied by the grip factors.
Latitudinal coefficient is in function of slip angle of the wheel. Slip angle is the angle represented by the wheel forward direction and the direction of motion of the vehicle.
Longitudinal coefficient is in function of slip rate, wich is delta RPM / max delta RPM we set before.

When changin the values use the template values as reference point.

Finishing up

Finally let's look into Aerodynamics Tab. You will find all the curves for the aerodynamic components.
Set the curve name and the coeficient. All curves are in function of Angle of attack. The angle of attack is the same as the tire slip. The angle formed between the forward wing direction and the direction the aircraft is moving. But in this case it is as seen from a side. Where is more pitch of the aircraft in relation to pitch of flight in local space.
For more information see Aerodynamics guide and tips from 1.2

Lastely look at systems and set the fuel unit weight and fuel tanks parameters.

The rest of the tabs are easy to follow and don't really requiere explanation.

If you want to use demo scene as testing platform, open the demo scene and look for _DemoMain object. It should have a DemoSceneManager component attached to it.
Expand available vehicles and add your own. Now run the demo scene and select your aircraft.

Using uSim vehicles.

To be able to use any uSim vehicle you need to have 2 things in scene. the inputs wrap classes called uSim_PlayerInterface. You can find the prefab located at Main/system prefabs folder.
Second you need to add the global atmoshperic values. Create an emtpy gameobject or attach the component to any object. The component is called “AtmosphericGlobals”
You can use fixed value or a value based in altitude.
Density factor is in funtion of altitude in mts * 10,000. Use demos scene parameters for confort.

USim also comes with support for rewired. Located in Main/scripts/inputs folder there's a package called RewiredSupportPack. Open it and you should get two new files in a new folder. One is the rewired configuration prefab, and the other is the script that has to be attached to uSim_PlayerInterface to support rewired inputs.

For more support don't hasitate to contact me at gabriel.campitelli@gmail.com

More updates will follow now that 1.3 is out.
Thanks for your support.