Distant Worlds 2

Making Ship Models

Version 1.54

March 2022

General Ship and Base Design Guidance

Below is some general guidance for model design, noting specific considerations for the various types of component bay.

1. The weapon slots are the most involved. Each has an intended weapon arc. All weapon sub-meshes should be either on the top or the sides of the ship and base hulls. The position and arcs are not an absolute requirement, but a strong design suggestion. We should get as close to this as possible.
    a. As the 3D ships will mostly be fighting on the same plane, weapons on the tops of ships should have a possible way within the suggested weapon arc to fire at a downward angle to intersect the horizon without firing through part of the ship hull.
    b. We leave exact positioning of the weapons and the design of the hulls to the concept and model artists, the slot locations on these symbolic hulls show our intentions but they are negotiable. We also want to end up with some unique and cool hull designs for the different factions and can adjust the design to make those possible if necessary.
    c. Weapons on the sides of ships may have the easiest time with access to the horizon, but some weapons due to their arcs will need to be on top of the ship.
2. There are two types of weapon slots that matter for art – Heavy and anything else. In the game the Heavy slots can support a weapon of any size, but could also be used for a smaller weapon. For those, we would like to have the option of a larger heavier weapon mesh that we can use to reflect Heavy weapons when they are placed there. Slots that do not have the option of a Heavy weapon do not need this.
3. There are two types of sensor slots that matter for art – Large and regular. These function similar to the Heavy weapon slots, in that a smaller sensor could be put in a Large slot, but when a Large sensor (such as a long range sensor, or one that coordinates targeting for an entire fleet) is placed there, we would like the option of a mesh that reflects a more impressive antenna/sensor dish/sensor array/etc. whatever we feel looks like an impressive sensor for that particular faction.
4. All ship models with engines should have an odd number of engines (total) so that the symmetry will work when the player designs a ship with not all engine slots used.
5. General slots can be ignored completely for art purposes.
6. The Approximate Size listed is an indication of the length and volume of each ship or base relative to others. For example, a Size 400 ship should be roughly twice as large as a Size 200 ship.
    a. A good rule of thumb is to take the Approximate Size and divide by 50 to determine a length for each ship or base model. Thus a cruiser that is size 400, would have a model around 8 units long.
7. The largest bases and ships are the most complicated. We suggest these to be left for last for each faction once the style is well established and modeling well underway.
8. The slot designs for the civilian ships are the same across all factions. However, they should not look the same at all in terms of look and style of the resulting models, even though they have the same sub-mesh requirements.

Coordinate system

For simplicity, all models should conform to a standard **right-handed coordinate system** where the following rules apply:

- Horizontal X axis aligns in a right-to-left direction, where values increase to the left (positive) and decrease to the right (negative)
- Vertical Y axis aligns in a bottom-to-top direction, where values increase to the top (positive) and decrease to the bottom (negative)
- Depth Z axis aligns in a back-to-front direction, where values decrease forward (negative) and increase backwards (positive). Note that this means **forward is negative-Z (0, 0, -1)**. This is after any compensating export rotations (e.g. to switch up-axis from Z to Y). Thus the default FBX export from 3ds Max rotates -90 degrees around the X-axis, **changing forward from positive-Y to negative-Z**

Exporting with Y-up

3D Modelling software often uses other coordinate systems, e.g. where Z is up. In these cases the export to the final FBX model must transform the coordinate system so that the correct axes are used, i.e. where Y is up.

Format of art assets

Although any 3D modelling tool can be used to create assets, the format of the final model assets should be as follows:

- All models should be **FBX** files
- All textures should be either **TGA** or **PNG** image files

Distant Worlds 2 uses Stride (https://www.stride3d.net/) to render ship models. Artists can use the free Stride Game Studio (https://www.stride3d.net/download/) to import and preview models and materials. This allows artists to check that the exported FBX model will appear as expected in the game. This can include testing animations using simple scripts.

Textures

Materials for ships and bases in Distant Worlds 2 will include the following texture layers:

- Diffuse for base color
- Normal map
- Emissive for lit windows, lights, engine glow, etc
- Combined texture for Metalness (Red channel), Glossiness (Green channel), Ambient Occlusion (Blue channel)

Textures for ships and bases should generally have a size of 2048 x 2048 pixels but should be created at the higher resolution of 4096x4096.

How Ships are designed in Distant Worlds 2

In Distant Worlds 2 all Ships and Bases are based on a Design.

Designs are like blueprints or templates that determine what a ship or base will be made of. Designs are a combination of a **Ship Hull** filled with **Components**.

Ship Hulls define which model and materials to use when rendering a ship. They also define the component bays in the hull. These component bays are used to slot in specific components.

Components are things like: weapons, engines, shields, fighter bays, etc. So to create a new Design you do the following:

1. Select a Ship Hull. The Ship Hull defines the Component Bays available to place components into
    a. Each Component Bay has a type and maximum size
2. Add Components to the Ship Hull in the provided Component Bays. Components typically need to match the type and size of the Bay
3. The completed Design can then be used as a template to make new Ships or Bases

Relationship of Ship Hulls and Components to Models

How do these game entities relate to models?

- Ship Hull = main ship mesh in ship model
- Components = each component is a separate mesh within the ship model

Not all components show up on the ship exterior: some are internal and have no visible representation. But exterior components that are visible on the ship include: engines, weapons, shields, hangar bays, sensors and antennae.

These exterior components are rendered in the game by component meshes in the ship model.

Component mesh enabling/disabling in Ship models

Because a ship design can have a variety of different components (as defined by the player when they design their own ships and bases) the ship models must support enabling/disabling meshes. This allows the correct component mesh (engine, weapon, hangar bay, etc) to be shown on the ship at the right position and rotation.

Mesh enabling means that the ship models must use a special naming convention for the component meshes, i.e. wherever an exterior component can be added in a ship design. This matches the Component Bays defined in the Ship Hull, as explained above.

For a more detailed description of how Ship Hulls, Components and Models relate to each other, see the section “Ship Hulls, Components and Models – in detail” at the end of this document.

Ship Models and Meshes

Each ship model should have a **single mesh** for the main section of the hull.

The ship model will also have **multiple additional component meshes** for exterior component bays, e.g. weapons, sensors, engines, hangars, etc

Component meshes

These are the component meshes that are enabled to show an actual component when the component bay is in use (e.g. engine, weapon, hangar bay, etc). When there is no component present in the bay, then they are not shown, i.e. their mesh is disabled.

Position and Rotation

Note that these **component meshes define the position and rotation** of any components that are added to the ship.

For example, the rotation of component meshes means the following:

- engines will face in the direction of the component mesh, with their exhaust effect rendered in this direction
- weapons will point out from the ship in the direction of the component mesh, and will rotate around this axis
- other ships and fighters will enter a hangar bay in the direction of the component mesh

Thus it is **critically important that the rotation for component meshes is accurate**, i.e. that they are facing in the proper direction.

Naming

Each component mesh should be named with a hash prefix, then the type followed by an index number, as shown below:

- Weapons are split into three separate categories, with potentially small and large versions for each category, as defined in each Ship Model layout. Thus there could be up to 6 weapon meshes at each weapon component bay position on the ship (when a heavy or large mount), or 3 weapon meshes if just a standard-sized mount. Each of these weapon meshes should be at the same position and rotation on the ship. The appropriate weapon mesh would be enabled when a weapon component is placed in the ship design for that bay, with all other weapon meshes at that position disabled.
    - **Barrelled Turret**: laser blasts, rail guns, point defense. Would likely use a mesh with barrel and spherical ball-like base that allows clean rotation in any direction
        - Standard size mesh names: “#weaponBarrelledTurret0”, “#weaponBarrelledTurret1”, “#weaponBarrelledTurret2”, etc
        - Large size mesh names: “#weaponBarrelledTurretLarge0”, “#weaponBarrelledTurretLarge1”, “#weaponBarrelledTurretLarge2”, etc
    - **Vertical Launcher**: missiles, torpedoes, assault pods. Would likely use a mesh with a grid of fixed vertical-launch style openings. **This type of weapon mesh does not rotate, instead the projectiles travel straight out of weapon openings for a short distance and then turn towards target**

        - Standard size mesh names: “#weaponVerticalLauncher0”, “#weaponVerticalLauncher1”, “#weaponVerticalLauncher2”, etc
        - Large size mesh names: “#weaponVerticalLauncherLarge0”, “#weaponVerticalLauncherLarge1”, “#weaponVerticalLauncherLarge2”, etc
    - **Short Turret**: phasers, tractor beams. Would likely use a mesh with a very short barrel and spherical ball-like base that allows clean rotation in any direction
        - Standard size mesh names: “#weaponShortTurret0”, “#weaponShortTurret1”, “#weaponShortTurret2”, etc
        - Large size mesh names: “#weaponShortTurretLarge0”, “#weaponShortTurretLarge1”, “#weaponShortTurretLarge2”, etc
- Engines: "#engine0", "#engine1", "#engine2", etc
    - also see notes on Engine Symmetry later in this document
- Hangar Bays: "#hangar0", "#hangar1", "#hangar2", etc
- Sensors are split into two separate sizes: standard and large. Thus there could be 2 sensor meshes at each sensor component bay position on the ship, depending on the size of the component bay as defined in each Ship Model layout. Both of these sensor meshes should be at the same position and rotation on the ship. The appropriate sensor mesh would be enabled when a sensor component is placed in the ship design for that bay, with the other sensor mesh at that position disabled
    - Standard size: typically dish-style mesh
        - "#sensor0", "#sensor1", "#sensor2", etc
    - Large size: typically antenna-style mesh
        - "#sensorLarge0", "#sensorLarge1", "#sensorLarge2", etc
- Defenses: "#defense0", "#defense1", "#defense2", etc

Support Structure meshes

These are optional supporting meshes that are only visible when a Component Bay is enabled, i.e. when it is in use. Support structure meshes are used to better position the component, or make it blend in better with the ship hull. These meshes are not required, but can be added to the ship model if needed.

A common example of a support mesh would be a weapon collar mount that is used to hold the weapon mesh.

If used, these support structure meshes should be named to match the corresponding component mesh, along with a "_support" suffix, as shown below:

- Weapons – All types, standard size: ''#weapon0_support", ''#weapon1_support", ''#weapon2_support", etc
    - Note however that because Vertical Launcher weapons do not rotate they also do not use weapon support meshes, thus the weapon support mesh will NOT be enabled when a vertical launch weapon is in the bay
- Weapons – All types, large size: ''#weaponLarge0_support", ''#weaponLarge1_support", ''#weaponLarge2_support", etc

    - Note however that because Vertical Launcher weapons do not rotate they also do not use weapon support meshes, thus the weapon support mesh will NOT be enabled when a vertical launch weapon is in the bay
- Engines: ''#engine0_support", ''#engine 1_support", ''#engine 2_support", etc
- Hangar Bays: "#hangar0_support", "#hangar1_support", "#hangar2_support", etc
- Sensors – Standard size: "#sensor0_support", "#sensor1_support", "#sensor2_support", etc
- Sensors – Large size: "#sensorLarge0_support", "#sensorLarge1_support", "#sensorLarge2_support", etc
- Defenses: "#defense0_support", "#defense1_support", "#defense2_support", etc

Hangar Bay support meshes

Note a special case usage of these support meshes for hangar bays: the hangar bay support mesh is used to cover the opening in the ship hull where the hangar bay mesh is located. When a hangar bay mesh is added to a component bay, and thus is shown on the ship, then the hangar bay support mesh is hidden, so that the hangar bay inside the ship is visible.

Thus instead of being **enabled** when a component is in the bay (as with all other component types), for hangar bays the support mesh is **disabled** when a hangar component is in the bay.

Thus hangar bays must always provide a support mesh – it is not optional.

Module meshes

In addition to component meshes, ship and base models may also have other meshes that can be enabled or disabled to give a different appearance to the final ship or base.

These module meshes are not related to component bays, and thus have no functional in-game effect. They are simply a cosmetic feature that changes the appearance of a ship. They allow for a modular approach to showing a ship, where different sections of a ship can be enabled/disabled to give it a different look.

Each module mesh should be named with a hash prefix, then “module” followed by an index number, as shown below:

- “#module0”, “#module1”, “#module2”, etc

The actual enabling/disabling of module meshes is controlled in the ShipHulls file (data/ShipHulls.xml). This allows a single ship model to be used for multiple in-game Ship Hulls that turn different modules/sections on or off for a different appearance. For example, a freighter ship model may have multiple cargo hold sections that can be turned on or off to make a larger or smaller freighter Ship Hull.

Note that disabled module meshes should never make a ship look like it is comprised of visually unconnected sections, e.g. a module removed from the middle of a ship model with no other connecting structure. Thus module meshes may work best at the extremities of a ship model, or as part of a larger superstructure that ‘contains’ them.

Rotating Module meshes

Module meshes can also be rotated to give an interesting visual effect on the ship or base. For example, a base model could contain a rotating outer-ring section.

The **default axis of rotation is defined by the rotation of the module mesh within the model** (in the same manner as component meshes). This means the difference from the default Z-axis. So if you want the module mesh to rotate around the Y-axis (up) then create the module mesh aligned with the Z-axis (rotating around the Z-axis), then rotate so that the rotation axis changes to the Y-axis (up).

Enabling rotation in-game will also require setting values in the ShipHulls file. The critical value here is the rotation rate. By default this is zero (no rotation), but can be set to any value in radians per second. The ShipHulls file also allows overriding the rotation axis if desired.

Other model features

Ship models can also have **running lights** that flash on/off on the ship or base. This can highlight the location of the ship, especially in a dark scene.

Also ship models can mark **points where particle effects will be emitted** when used in the game. This allows precise positioning of animated effects like smoke/dust from mining, sparks/welding from construction, etc.

Running lights and emitters are not related to Component Bays or Modules. But they do use a similar system of mesh naming. A simple named mesh can be placed on the ship model wherever a running light or emitter must be positioned. In the game when the model is loaded, the position of these meshes will be recorded, **but the meshes themselves will be removed** – they are just placeholders. Running lights and emitters can then use these marked positions.

Note that unlike component meshes, running light and emitter meshes do NOT indicate direction in any manner, thus their rotation is not important, just their position on the model.

These meshes should be named with a hash prefix, then the type followed by an index number, as shown below:

- Running lights: #runninglight0, #runninglight1, #runninglight2, etc
- Emitters: #emitter0, #emitter1, #emitter2, etc

Special considerations for Engine Symmetry

There is a special requirement to consider for engine meshes. The final ship model should always maintain symmetry with the engine components, i.e. the number and placement of the engines on the model should be symmetrical, so that there is the same number of engines on each side of the length-wise axis (Z-axis).

Because the player can design their ships with any number of engine components, there needs to be a standard convention for engine meshes to ensure engine symmetry.

This convention has the following rules:

- All ship models must have an **odd number** of engine meshes

    - This allows any number of engine components to be added and still achieve engine symmetry, e.g. 1 engine will use the center position, 2 engines would be using the first paired positions (as described below), 3 engines would use the center position and the first paired positions, etc
- The first engine mesh (#engine0) must be in the centerline of the length-wise axis (Z-axis)
- All subsequent engine meshes must be paired symmetrically on each side of the length-wise axis (Z-axis)
    - thus #engine1 and #engine2 are paired symmetrical positions on opposite sides of the model
    - then #engine3 and #engine4 are also paired symmetrical engines on opposite sides of the model, as are #engine5 and #engine6, etc

Mirroring Components

As most ships will be symmetrical along at least one axes, it is very tempting to create one side's components and mirror them across to complete the other side. Using this technique within 3dsMax 2015 has led to problems with normals being flipped on the mirrored objects after conversion to an FBX. Applying a "resetXform" followed by a "Normal" modifier solved the issue.

Ship Model naming

All ship models should simply be named after their ship role, e.g. "cruiser", "frigate", "miningship", etc. A full list of all ship roles is shown below:

- Escort
- Frigate
- Destroyer
- Cruiser
- CapitalShip
- Carrier
- TroopTransport
- FreighterSmall, FreighterMedium, FreighterLarge
- MiningShip
- ConstructionShip
- ExplorationShip
- PassengerShip
- ColonyShip
- SpaceportSmall, SpaceportMedium, SpaceportLarge
- MiningStation
- DefensiveBase
- ResearchStation
- MonitoringStation
- ResortBase
- FighterInterceptor, FighterBomber
- FuelTanker
- PlanetDestroyer

Ship Textures

Textures should be named after the ship role and a suffix indicating the texture type, as follows:

- Diffuse: "cruiser_diffuse"
- Normal: "cruiser_normal"
- Emissive: "cruiser_emissive"
- Combined Metalness, Glossiness, Ambient Occlusion: “cruiser_pbr”

Ship Hulls, Components and Models – in detail

To support component mesh enabling/disabling in models we have the following data for Models, Components and Ship Hulls:

- **Ship and Base Models**
  - single mesh for main body of ship or base
  - multiple additional component meshes for exterior component bays with visible components, e.g. weapons, sensors, engines, hangars, etc
    - these meshes follow a **special naming convention** (as outlined elsewhere in this document) to allow programmatic inspection and identification of these parts, along with their **position** and **rotation**.
    - if an exterior Component Bay is unused in a Design then the component mesh for that bay is simply made invisible (i.e. turn off rendering for that mesh), showing whatever is underneath that mesh in the main body of the model
      - Hangar Bays are an exception: for unused hangar bays we instead render the **support mesh**, which should cover the hangar bay opening in the main hull of the ship model
    - each exterior component bay can also have an additional '**supporting structure**' mesh that is only visible when the bay has a component, i.e. when a component mesh is enabled on the ship at that position.
      - the supporting structure mesh can be used to better position the component, or make it blend in better with the ship hull. Examples include:
        - 'collar' for weapon mounts
        - support structure for engines
- Individual ship **Components** define:
  - which type of component mesh to use (e.g. weapon turret, engine nozzle, sensor dish, shield projector or hangar bay). Components only ever have a single mesh, with some important implications outlined in Weapon Turrets section below...
  - size of the component
  - type of rotation for the component:
    - None – static, no movement
    - Rotate to target (e.g. direct fire weapon)
    - Constant rotation (e.g. sensor dish)
  - rate of rotation, e.g. how fast a weapon turret rotates to target, how fast a sensor dish rotates
- **Ship Hulls** define:
  - all of the Component Bays for the hull. Each Component Bay defines:
    - the type of bay (weapon, defense, engine, sensor, hangar, general)
    - the name of the mesh in the model that maps to the bay. This only applies to exterior bays on the model, e.g. weapons, engines, etc
    - the maximum size of the component bay, defining which components can fit in the bay. This can also alter the selection of which component mesh is used to render the component on a ship, e.g. if weapon or sensor

component is large than large versions of these component meshes will be used.
        - position of the mesh in the model - read directly from model itself, but can be manually overridden
        - directional 3D vector that defines 'Up' for the mesh, i.e. pointing directly out from the model. This vector is determined by the rotation of the component in the model itself. This vector is used in the following ways:
            - Sensors: this is the axis of rotation for all components with a 'constant' rotation type, e.g. sensor dishes
            - Weapons: for all components with a 'rotate to target' rotation type this is used as center of the weapon firing arc/hemisphere. For seeking weapons this is used as the initial firing direction of the missile/torpedo
            - Engines: direction of engine exhaust effect
            - Hangar bays: direction of entry to hangar bay for ships and fighters
        - Range of weapons arc, i.e. the angle of firing arc centered on the directional 3D vector above. This defines the size of the 'hemisphere' of coverage for the weapon. Typically this would be around 180 degrees, allowing the weapon to fire on half the sky
    - which Module meshes are enabled/disabled (if any exist in the model), along with the rotation rate for each module (if rotating)

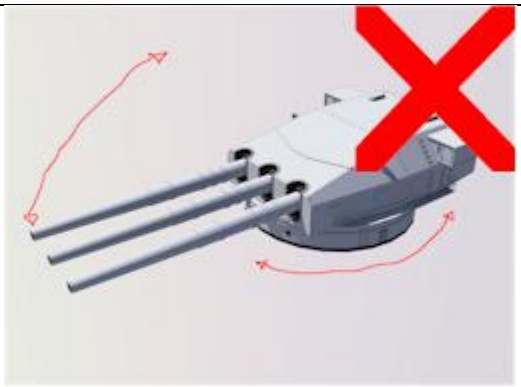
Thus a finished Ship or Base Design would actually construct a finished model with all component meshes enabled/disabled to give the final look for that Design, with the desired weapon, engine, sensor and hangar meshes for the Design. This finished model is then used when rendering ships or bases that are based on the design in the game.

Finally, some extra notes on specific cases:

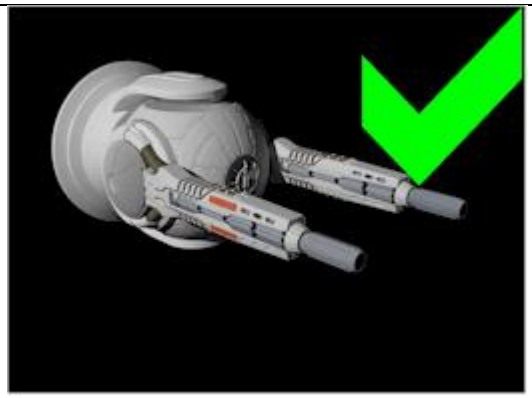
Weapon Turrets

There is an **important constraint** to note here: components must be comprised of only a single mesh, i.e. they have no moving parts within themselves, e.g. weapon turrets do not have separate turret and barrel - they rotate as a single part. Thus the whole weapon turret mesh will be rotated, so the shape of the mesh may be best as a sphere with barrel, or maybe a cylinder with a barrel protruding from one side. This is because the sphere or cylinder part will be rotated in all dimensions along with the barrel - there is no separate turret rotation and barrel elevation.

However support structure meshes (described above) can be used as ‘collars’ to blend the weapon into the main ship model.



NOT LIKE THIS: Weapon turret with separate rotation for turret and elevation for barrels - not possible in DW2



MORE LIKE THIS: Weapon turret that can cleanly rotate as single unit (i.e. spherical portion and barrels together)

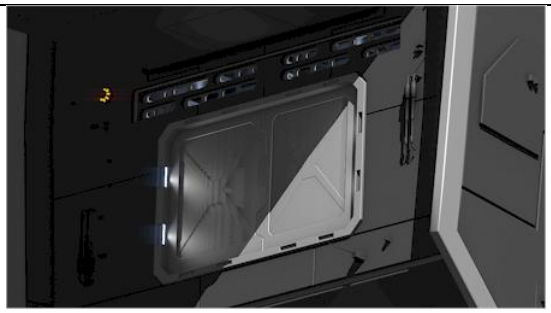
Hangar bays

These are internal bays that sink into a ship or base, with an external opening flush with the surface of the ship/base.

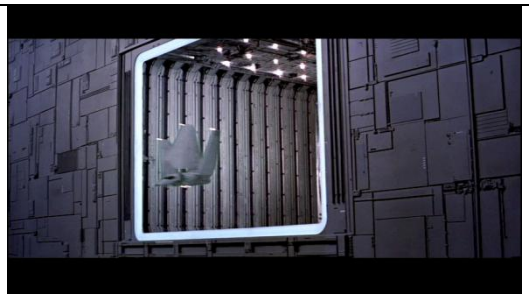
When a hangar bay is not used on a ship, then the opening in the hull of the ship model must be covered. The **hangar support mesh** is used to cover an unused hangar bay. Thus the support mesh for a hangar bay might just be a flat surface that blocks off the entrance to the hangar, i.e. when it's not in use, no hangar component enabled in the ship design. Then when a hangar component of some sort is added (fighter bay, docking bay, construction bay) to this Component Bay, the support mesh is disabled and the hangar bay component mesh is enabled. This would be sunken into the model, giving a concave view inside.

Thus in the main body of the ship model, at the location of a hangar bay mesh the model would actually have an opening (i.e. no vertices) or hole that would be covered by the flat hangar support mesh.

When the concave hangar bay mesh is in use, the covering support mesh would then be disabled, allowing an unobstructed view inside the hangar.



Possible view of Unused hangar bay - flat covering support mesh flush with surface, but maybe with edge detail showing location of potential hangar



Hangar bay in use – disable support mesh and enable hangar mesh that sinks into ship/base, allowing view inside