Unity → Three.js Asset Export Guide

For: AI Agent with access to the Ocean Outlaws Unity project

> **Goal:** Extract ship and environment models from the Palmov Island Unity asset packs, optimize them for web, and deliver clean GLB files ready for use in the Ocean Outlaws Three.js renderer.
>
> **Input:** Unity project with both asset packs installed
> **Output:** Optimized `.glb` files in `/exports/optimized/`

Agent Checklist Overview

```
[ ] Stage 1 — Verify Unity project state
[ ] Stage 2 — Install UnityGLTF exporter (if not present)
[ ] Stage 3 — Upgrade materials to URP
[ ] Stage 4 — Prepare and export each model as GLB
[ ] Stage 5 — Install gltf-transform and optimize all GLBs
[ ] Stage 6 — Validate outputs at target specs
[ ] Stage 7 — Confirm sail mesh naming convention
```

Stage 1 — Verify Unity Project State

Before doing anything, confirm the project is healthy.

Check 1 — URP is active:

- Open `Edit → Project Settings → Graphics`
- Confirm the Scriptable Render Pipeline asset is a URP asset (not blank, not HDRP)

Check 2 — Both asset packs are installed:

- In the Project panel, confirm these folders exist:
  - `Assets/LowPolySeaLocationsPack/` (or similar — Palmov Island environment pack)
  - `Assets/LowPolyCartoonSailingShips/` (or similar — Palmov Island ships pack)
- If either is missing, open `Window → Package Manager → My Assets` and import the missing pack

Check 3 — No compiler errors:

- Bottom status bar should be clear. If there are errors, resolve them before proceeding — broken projects produce broken exports.

Stage 2 — Install UnityGLTF Exporter

Unity has no native GLB export. UnityGLTF is the standard solution.

1. Open `Window → Package Manager`
2. Click `+` → `Add package from git URL`

3. Enter:

   ```
   https://github.com/KhronosGroup/UnityGLTF.git
   ```

4. Click **Add** and wait for compilation to finish
5. Verify installation: top menu bar should now show a **GLTF** menu item

> If the project already has UnityGLTF installed (check `Packages/manifest.json` for `"com.unity.gltf"`), skip this stage.

Stage 3 — Upgrade Materials to URP

Palmov packs may include materials authored for the Built-in pipeline. These appear as **pink/magenta** in the scene if not converted.

1. Go to `Edit → Rendering → Render Pipeline → Universal Render Pipeline → Upgrade Project Materials to UniversalRP Materials`
2. Click **Proceed** in the confirmation dialog
3. After conversion, do a visual scan of the ship prefabs and environment prefabs in the scene — all materials should show correct flat colors, not pink

> This is a one-time operation. It modifies material assets in place. If it has already been run, materials will already be correct — skip if everything looks fine visually.

Stage 4 — Export Models as GLB

Create a clean working scene, then export each asset one at a time.

4.1 Create an Export Scene

1. `File → New Scene` → choose **Empty**
2. Save it as `Assets/Scenes/ExportScene.unity`
3. Delete any default directional light or camera — they are not needed

4.2 Export Each Model

For **each asset in the export list below**, repeat this process:

1. Locate the prefab in the Project panel (see paths in the export list)
2. Drag it into the Hierarchy
3. Set Transform in the Inspector: Position `(0, 0, 0)`, Rotation `(0, 0, 0)`, Scale `(1, 1, 1)`
4. Select the root object in the Hierarchy
5. Go to `GLTF → Export selected as GLB`
6. In the export dialog:
   - Format: **GLB**
   - ✅ Export Textures
   - ✅ Export Materials

- ✅ Export Children (preserve hierarchy)

7. Save to `/exports/raw/` using the exact filename from the export list
8. Delete the object from the Hierarchy before adding the next one (keep the scene clean)

4.3 Ship Export List

Output Filename	Source Prefab (look in the ships pack folder)
`ship_sloop.glb`	Smallest single-mast vessel
`ship_brigantine.glb`	Two-mast mid-size ship
`ship_galleon.glb`	Large three-mast vessel
`ship_manowar.glb`	Largest, most decorated ship

> The exact prefab names will vary by pack version. Browse `Assets/LowPolyCartoonSailingShips/Prefabs/` and match visually. When in doubt, export the one that fits the size/class description.

4.4 Environment Export List

Output Filename	Source Asset (look in the locations pack folder)
`env_island_tropical.glb`	Tropical island with palm trees
`env_harbor.glb`	Port or dock structure
`env_sea_cave.glb`	Rock arch or sea cave formation
`env_shipwreck.glb`	Wrecked/sunken ship asset
`env_lighthouse.glb`	Lighthouse tower
`env_rocks.glb`	Rock cluster / sea stack

> If the pack uses scene files rather than individual prefabs, identify each element in the demo scene, isolate it in the Hierarchy, and export just that selection.

Stage 5 — Sail Mesh Naming Convention

This must be done before exporting — it enables runtime faction recoloring in Three.js.

For each ship prefab, before running the GLB export:

1. Expand the prefab hierarchy in the Hierarchy panel
2. Find all mesh objects that represent sails (usually named `Sail`, `Sail001`, `SailMesh`, etc.)
3. Rename each sail mesh to follow this convention:
   - `sail_main`
   - `sail_fore`
   - `sail_mizzen`

- (add `_01`, `_02` suffixes if there are multiples of the same type)

4. Hull, deck, and rigging meshes should **not** contain the word `sail` in their names

> Why this matters: The Three.js loader traverses the model and recolors any mesh whose name contains `sail`. Consistent naming here prevents hull meshes from being accidentally recolored and ensures faction colors work at runtime.

Stage 6 — Optimize with gltf-transform

Raw Unity GLB exports are often oversized. This stage compresses them for fast web loading.

6.1 Install gltf-transform (requires Node.js ≥ 16)

```
npm install -g @gltf-transform/cli
```

Verify install:

```
gltf-transform --version
```

6.2 Optimize Each File

Run this command for each GLB in `/exports/raw/`:

```
gltf-transform optimize exports/raw/ship_sloop.glb exports/optimized/ship_sloop.glb --compress draco
```

Or batch process the entire folder:

```
mkdir -p exports/optimized
for f in exports/raw/*.glb; do
  filename=$(basename "$f")
  gltf-transform optimize "$f" "exports/optimized/$filename" --compress draco
done
```

6.3 Target Specs

After optimization, each file should meet these targets:

Metric	Target
File size — ships	< 150 KB each
File size — environments	< 250 KB each
Triangle count — ships	< 5,000 per model
Triangle count — environments	< 8,000 per model
Textures	Flat color only; no large texture atlases expected

6.4 Validate Each Output

Drag each optimized GLB into **https://gltf.report** and confirm:

- Model renders correctly (no missing geometry, no pink materials)

- Triangle count is within target
- Sail meshes are named correctly and visible as separate objects in the scene tree

---

Stage 7 — Final Output Structure

When complete, the `/exports/` directory should look like this:

```
exports/
├── raw/                        ← Unity exports, unoptimized (keep as backup)
│   ├── ship_sloop.glb
│   ├── ship_brigantine.glb
│   ├── ship_galleon.glb
│   ├── ship_manowar.glb
│   ├── env_island_tropical.glb
│   ├── env_harbor.glb
│   ├── env_sea_cave.glb
│   ├── env_shipwreck.glb
│   ├── env_lighthouse.glb
│   └── env_rocks.glb
│
└── optimized/                  ← Final deliverables for Three.js
    ├── ship_sloop.glb
    ├── ship_brigantine.glb
    ├── ship_galleon.glb
    ├── ship_manowar.glb
    ├── env_island_tropical.glb
    ├── env_harbor.glb
    ├── env_sea_cave.glb
    ├── env_shipwreck.glb
    ├── env_lighthouse.glb
    └── env_rocks.glb
```

Once the `optimized/` folder is complete, copy its contents into the Ocean Outlaws Three.js project at:

```
ocean-outlaws/game/assets/models/
```

Troubleshooting Reference

Problem	Cause	Fix
Pink/magenta materials	Built-in pipeline materials not converted	Re-run Stage 3 URP material upgrade
GLB exports with no textures	Export dialog option unchecked	Re-export with “Export Textures” checked
Model rotated 90° in Three.js	FBX source bakes a	Before export, apply rotation in Unity: select mesh, `Ctrl+A` → apply all transforms. Or fix in code with `model.rotation.x = -Math.PI / 2`

	rotation	
GLTF menu not visible	UnityGLTF not installed or failed	Check `Packages/manifest.json` for the package; reinstall if missing
Draco decode error in browser	Three.js missing Draco decoder	Set decoder path in Three.js: `dracoLoader.setDecoderPath('https://www.gstatic.com/draco/versioned/decoders/1.5.6/')`
File too large after optimization	Source mesh too dense	Check triangle count in gltf.report; if over target, the source prefab may have LOD variants — use the lowest LOD
Sail mesh not recoloring at runtime	Mesh not named with `sail` prefix	Go back to Unity, rename the sail child mesh, re-export
Missing pack folder in Project panel	Pack not imported	Open Package Manager → My Assets → import the missing pack

---

Three.js Loading Reference

Once assets are in place, this is the loading pattern used in Ocean Outlaws:

```
import { GLTFLoader } from 'three/examples/jsm/loaders/GLTFLoader.js';
import { DRACOLoader } from 'three/examples/jsm/loaders/DRACOLoader.js';

const dracoLoader = new DRACOLoader();
dracoLoader.setDecoderPath('https://www.gstatic.com/draco/versioned/decoders/1.5.6/');

const loader = new GLTFLoader();
loader.setDRACOLoader(dracoLoader);

// Load ship model
loader.load('/assets/models/ship_sloop.glb', (gltf) => {
  const ship = gltf.scene;
  scene.add(ship);
});

// Recolor sails for faction (called after load)
function setFactionColor(shipScene, hexColor) {
  shipScene.traverse((child) => {
    if (child.isMesh && child.name.toLowerCase().includes('sail')) {
      child.material = child.material.clone();
      child.material.color.set(hexColor);
    }
  });
}

// Faction color constants
const FACTION_COLORS = {
  player:   0xf5f0dc, // Cream/natural
  pirates:  0xff2200, // Red
```

```
  navy:     0x1a3a6b, // Navy blue
  merchant: 0xc8a850, // Gold/tan
};
```

Guide version 1.0 — Ocean Outlaws pirate rework, Unity → Three.js asset pipeline