第３回　ジオ・コスモス コンテンツ コンテスト
技術説明会セッション１　資料

Unity を使った球体ライブコンテンツの制作方法紹介

改訂 1.1 版

2017 年 1 月 7 日・14 日

日本科学未来館

Miraikan

概要

この資料では、Unity®[1]を用いてジオ・コスモス用のライブコンテンツ[2]を開発する方法を紹介します。

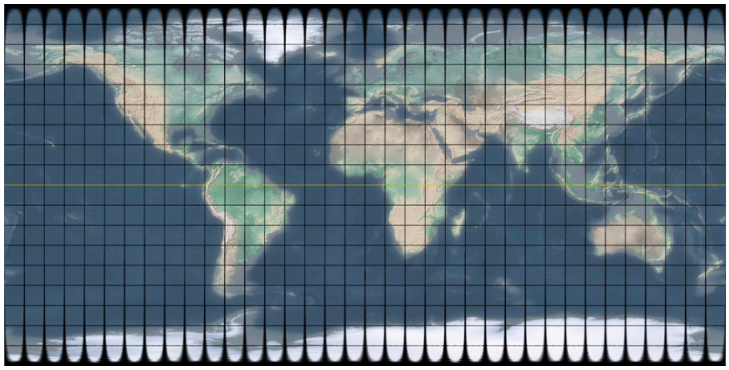
ジオ・コスモスには、あらかじめ収録された映像を再生するだけでなく、外部から入力した映像をそのまま表示する機能があります。ライブコンテンツを表示する場合には、この外部入力機能を用います。

外部入力できる映像信号の形式は以下の通りです。

解像度	4K（3840 x 2160 ピクセル）
リフレッシュレート	30p

映像信号は、HDMI ケーブルを介してジオ・コスモスのシステムに入力します。

入力する映像は、ジオ・コスモスに映す球面を**正距円筒図法で表現したフルスクリーンの映像**です。正距円筒図法は地図投影法の一つで、下の図のように緯線と経線が全て直交し、角度は等間隔に刻まれていることが特徴です。



正距円筒図法で描かれた世界地図

正距円筒図法の画像の比率は 2：1 ですが、ジオ・コスモスに入力する映像の比率は 16：9 なので、実際には若干縦方向に引き伸ばした映像を入力することになります。

[1] Unity は Unity Technologies の商標です。
[2] 双方向性のあるコンテンツやデータをリアルタイムに可視化するコンテンツなど、PC 等で生成した画像をリアルタイムで表示するコンテンツを、ここではライブコンテンツと呼んでいます。

ここからは、チュートリアル用のコンテンツに沿って説明していきます。

コンテンツは Unity パッケージ形式になっています。Unity で空のプロジェクトを作成し、メニューの **Assets > Import Package > Custom Package** から、以下のパッケージ[3]を選んでインポートしてください。

GC3TechnicalBriefingMaterials.unitypackage

●チュートリアル用コンテンツの概要

/GC3 Technical Briefing/Session1 Unity Tutorial/

チュートリアル本体が入っているフォルダです。チュートリアルは 6 つのシーンファイルとして用意されています。

1. **Rendering whole sphere with 6 cameras.unity**
2. **Rendering the Earth with 6 cameras.unity**
3. **Using Equirectangular Script.unity**
4. **Lightings and Settings.unity**
5. **Object Order and Polygon Face.unity**
6. **Try to build your Scene.unity**

上記のチュートリアルシーンで使われているマテリアルやスクリプト等の素材は以下のサブフォルダに置かれています。（後述の EquirectangularCamera スクリプト関連の素材を除く）

.../Tutorial Resources

チュートリアルで使用した素材の中で、特にジオ・コスモスのコンテンツを制作する際に便利と思われるものを選んで以下のフォルダに入れてあります。

.../Tutorial Resources/Useful Resources

オリジナルコンテンツを制作する際に自由にお使いいただいて構いません。なお、この中の世界地図関連の素材は、NaturalEarth[4] が提供しているデータ、またはそれを加工した素材です。

[3] プロジェクト全体をアーカイブした GC3Tutorial.zip もありあす。Unity に読み込まずに内容を見たい方はこちらを使用してください。

[4] http://www.naturalearthdata.com

/Q-Tools/EquirectangularCamera/

Unity を使って、全球が正距円筒図法で表現された映像を簡単に生成できるスクリプトです。Camera オブジェクトにアタッチするだけで、そのカメラを中心とした全球が正距円筒図法で出力されます。このスクリプトは、オリジナルコンテンツの制作に自由にお使いいただいて構いません。

/Q-Tools/Demonstration/

技術説明会のセッション 2「Geo-Cosmos のシステム紹介とデモンストレーション」で紹介した、ジオ・コスモスの表現特性を知っていただくためのデモコンテンツです。

Miraikan

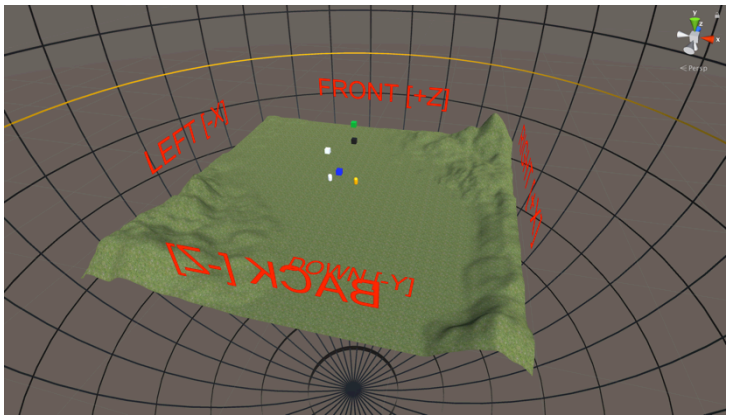
チュートリアル#1

Unity のシーンを全球でレンダリングする方法

シーン「1. Rendering whole sphere with 6 cameras」を参照してください

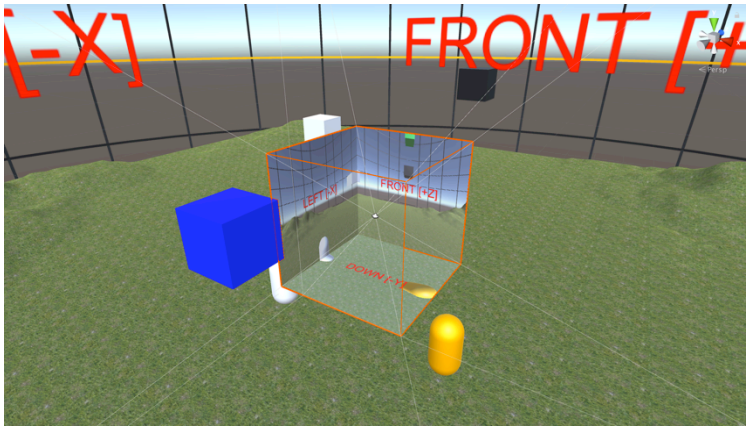
チュートリアル＃1、＃2 では、Unity 内につくられた 3D のシーンで全球画像を生成する方法について、その考え方を説明します。具体的には、6 つのカメラで CubeMap を作り、正距円筒図法に合成する方法です。一般的な方法なので、既にご存知の方は＃1、＃2 は読み飛ばしていただいて構いません。

チュートリアル＃1 では、以下のようなシーンを用います。



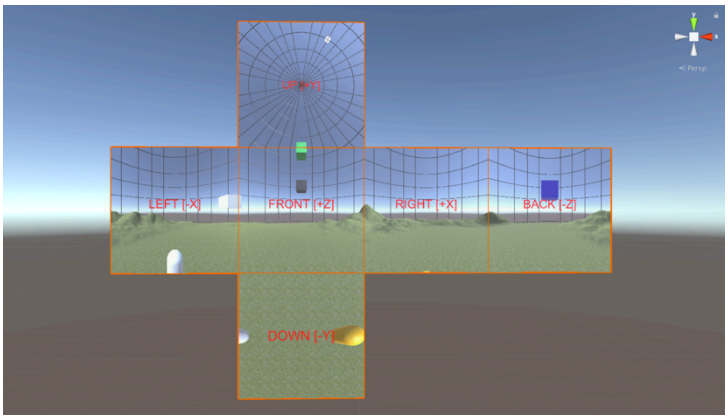
効果がわかりやすいように前後左右上下を示したテキストと、10 度刻みの角度目盛りを周囲に配置しました。

このシーンの中央から見た周囲全体を正距円筒図法で表現した画像はどのようにしたら得られるでしょうか。ひとつの Camera で周囲全体を一度にレンダリングすることはできません。そこで、前後左右上下の画像をそれぞれ個別に撮影し、あとでつなぎ合わせて全周の画像に合成することを考えます。Unity では、前、後、右、左、上、下に向けた、視野角 90 度で正方形に切り取るカメラを 6 台用意して、シーンの中央に置くことで、それらの画像を得ることができます。この様子を示したのが次の図です。6 つの方向を向いたカメラと、それぞれのカメラがどんな画像を捉えているかを表しています。(描かれていませんが、実際は上、右、手前の画像も存在しています。)



視野角が 90 度なので、各画像の境目は綺麗につながっていて、各カメラが周囲の世界を重なりなく、もれもなく撮影できていることがわかります。

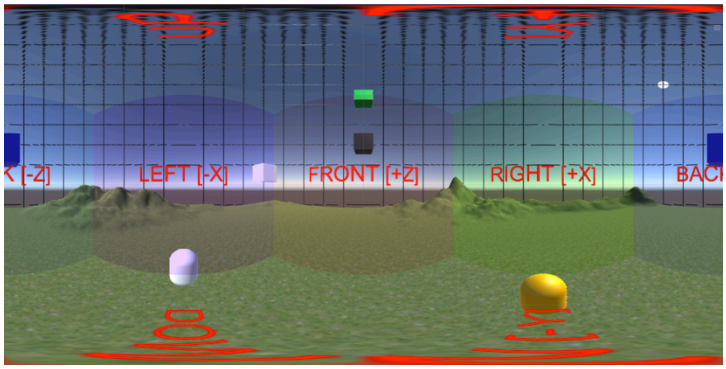
こうして得られた 6 枚の画像を、正方形の展開図として平面に並べたものが以下の図です。



各画像のつなぎ目は綺麗につながっていますが、緯線はつなぎ目で折れ曲がって直線にはならず、また上と下の画像は正面の画像としかつながっていません。正距円筒図法の画像にするためには、緯線をまっすぐにし、上下の画像も前後左右の 4 枚とすべて繋がるように、各画像を少しずつ歪めて再構成する必要があります。

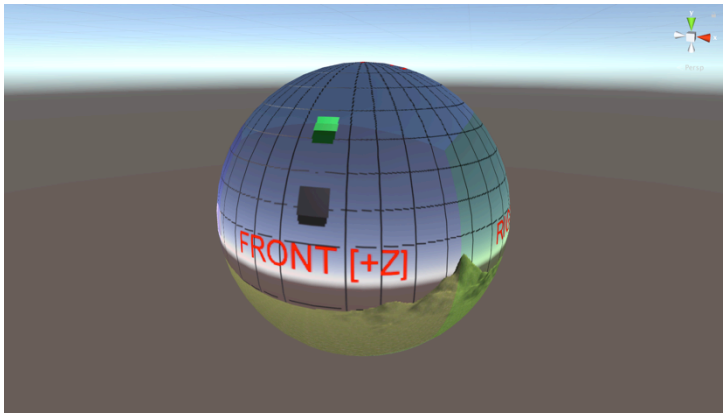
正しい歪みでつなぎ合わせたもの[5]が次の図です。（画像の境界がわかりやすいように、各面を薄く着色してあります。）

[5] 画像のつなぎ合わせには専用のシェーダーを使っています。興味のある方は、/GC3 Tutorial/Tutorial Resources/Materials/Equirectangular Projection/CubeFacesToEquirectangular.shader を参照してみてください。（ただし、原理説明用のシェーダーなので、実用には向きません。）



緯線も経線も直線として描かれ、等間隔に並んでいます。また、上や下の端の方は横方向に大きく引き伸ばされて、経線が太くなっていることがわかります。

こうして得られた画像をテクスチャーとして球体にマッピングしたものが以下の図です。画像が歪みなく繋がり、緯線、経線も正しく描かれているのがわかります。



ジオ・コスモスに入力された正距円筒図法の映像も、同様の方式でマッピングされますので、画像を球体にマッピングするという方法は、制作したコンテンツをプレビューする手段としても利用できます。

Miraikan

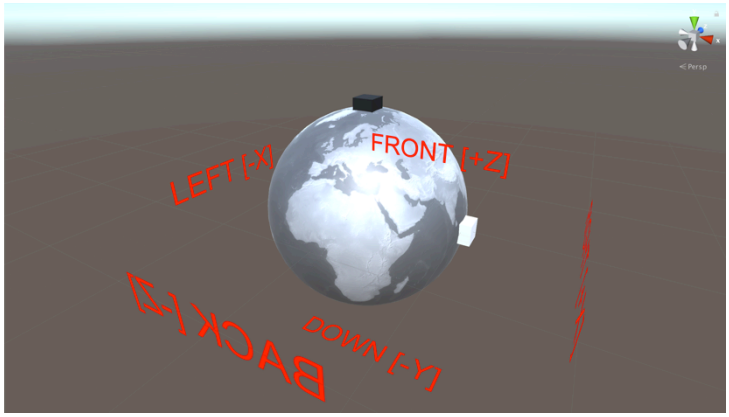
チュートリアル#2

球面上につくられたシーンを全球でレンダリングする方法

シーン「2. Rendering the Earth with 6 cameras」を参照してください

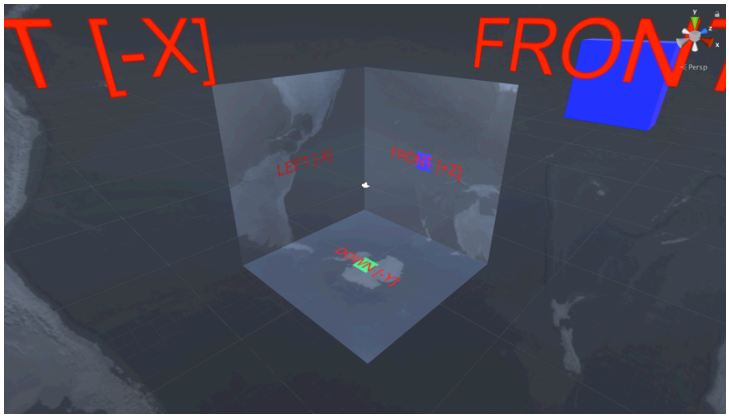
前のチュートリルでは、中心から見た全周の光景を正距円筒図法で表現しました。しかし、ジオ・コスモスのコンテンツのような球体に表示する映像を制作する場合、中心から見た光景ではなく、球体を外から見た状態を正距円筒図法で表現したくなります。

チュートリアル＃2 では、以下のようなシーンを使って、外側から見た様子を正距円筒図法に変換する方法について考えます。

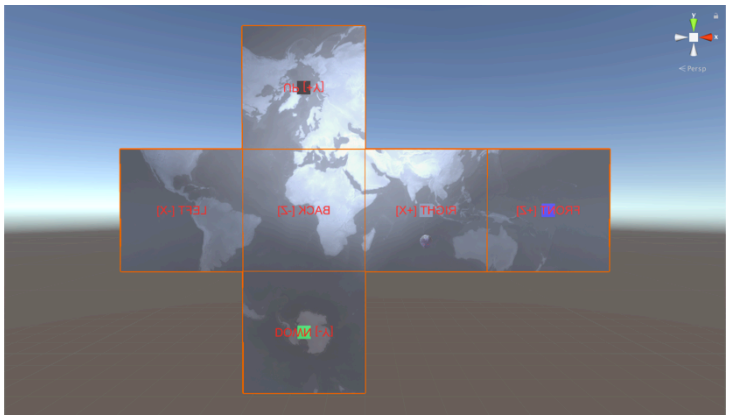


シンプルな地球儀の周りを、いくつかのオブジェクトが動き回っているシーンです。先と同様、分かりやすいように前後左右にラベルを置きました。今回は外側から見た様子なので、BACK[-Z]と書かれた側（経度０度の側）が中央になるような正距円筒図法の映像が得たいと思います。

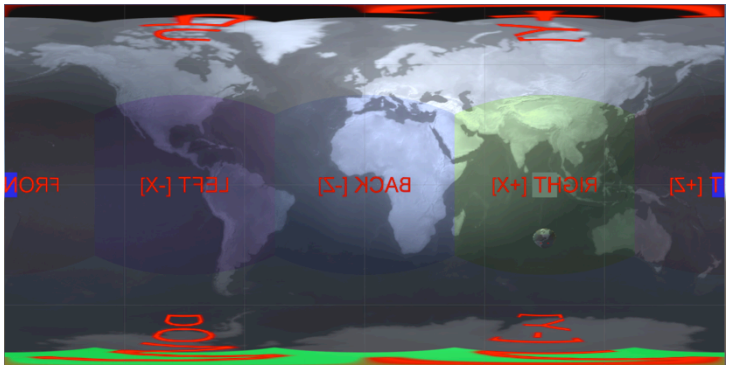
このシーンでも、地球儀の中心に６つのカメラが置かれていて、全球が撮影できています。しかし、内側から見ているため、（当然ですが）大陸の形が鏡像になっています。



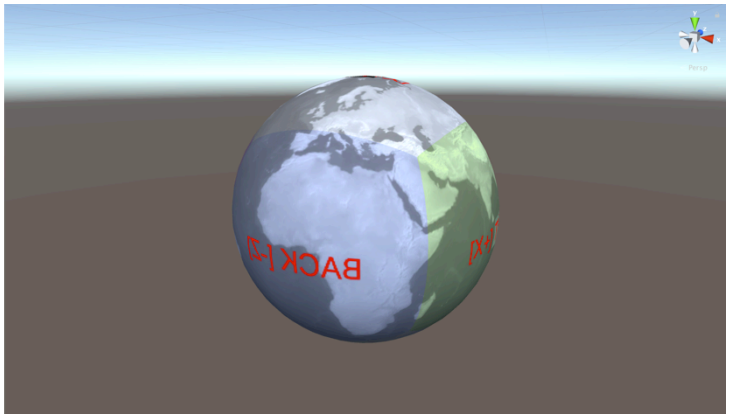
そこで、展開図の形に並べる際に、各画像の左右を反転させ、BACK[-Z]の面が中心なるように並べ方を変えました。



これらの画像を、チュートリアル＃１と同じ方法で、正距円筒図法に合成したものが次の図です。



これを球面にマッピングすると、期待通りの地球儀を表現することができました。チュートリアル＃１と違うのは、展開図を作るときに画像を反転させていることと、並べ方を変えたことだけで、他は全く同じです。



Miraikan

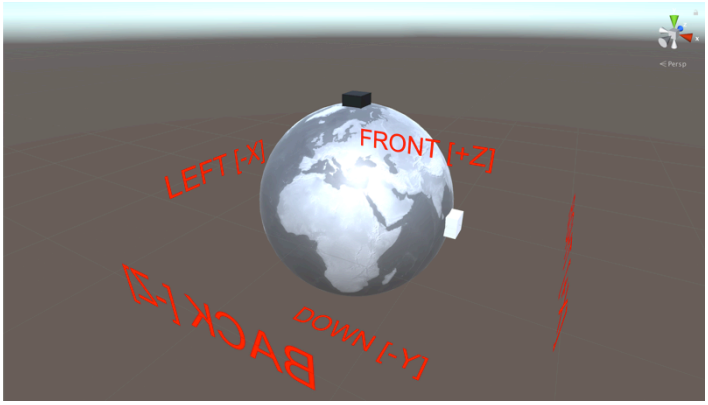
チュートリアル#3

簡単なスクリプトで正距円筒図法を生成する

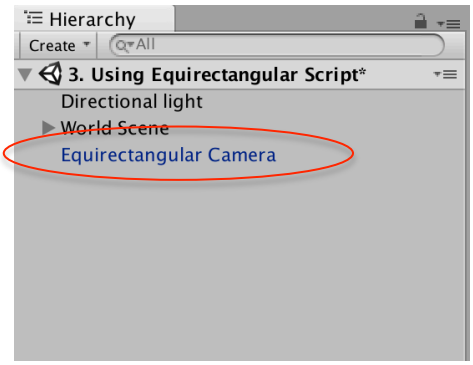
シーン「3. Using Equirectangular Script」を参照してください

ここまでは、正距円筒図法で表現された全球画像を作る方法について、考え方を説明してきました。シーンに実際にカメラを 6 台配置して合成する流れを見てきましたが、それらをまとめて行うスクリプトを用意しましたので、ここからはそのスクリプトを使っていきます。

チュートリアル＃3 に用意したのは、先と同じ以下のようなシーンです。

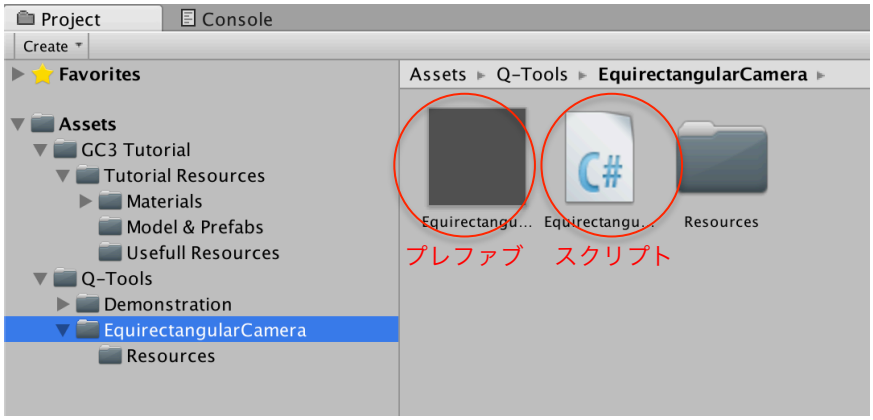


このシーンに配置されたオブジェクトを見ると、6 台のカメラは無く、**Equirectangular Camera** というプレファブが置かれているだけです。

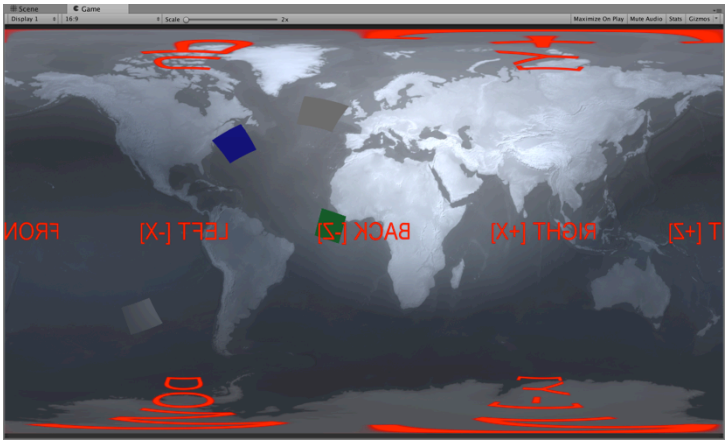


このプレファブは、EquirectangularCamera.cs というスクリプトがアタッチされた Camera オブジェクトです。このカメラでレンダリングされる画像は、その位置から見た全球の様子を、（チュートリアル＃2 と同様の）**外側から見た向きで正距円筒図法にした画像**になります。

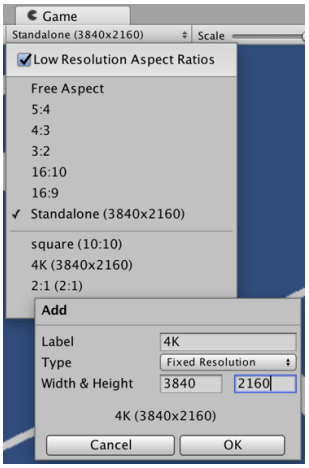
スクリプトとプレファブは、**/Q-Tools/EquirectangularCamera** というフォルダにあります。



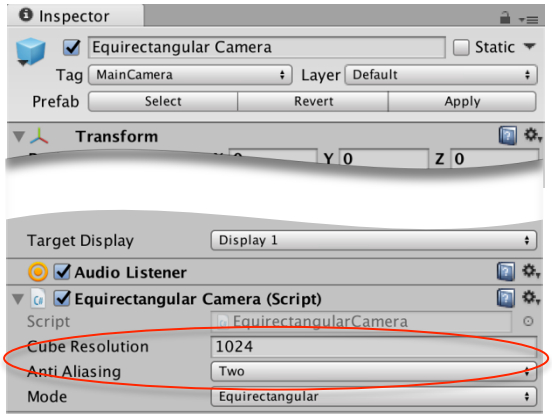
ゲームビューには、Play 状態でなくても正距円筒図法で描かれた画像が表示されます。



エディターで表示される画像サイズはゲームビューの設定に従うので、16：9、または、本番時の画質や処理速度を確認したいときには、明示的に 3840x2160 を指定すると良いでしょう。



EquirectangularCamera.cs スクリプトには、画質に関係するパラメータが二つあります[6]。



Cube Resolution：

6 つのカメラでレンダリングする各面の解像度（正方形の一辺のピクセル数）です。2 の累乗の値を指定してください。この値は処理負荷に大きく影響します。

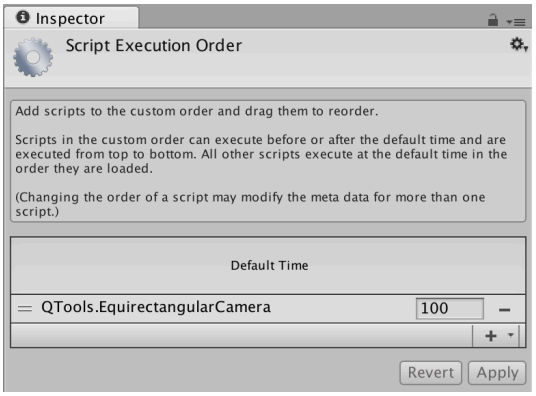
Anti Aliasing：

6 つのカメラでレンダリングする時のアンチエイリアスの程度です。数値が大きいほど、生成される画像は滑らかになります。処理負荷への影響は解像度よりは穏やかです。

【少し踏み込んだ話】

このスクリプトは、6 台のカメラで撮影するのと同じことを **LateUpdate()**の中で行います。そのため、他のスクリプトの **LateUpdate()**呼び出しより後で呼ばれるように呼び出し順を遅く指定してあります。

（メニューの **Edit > Project Settings > Script Execution Order**）



[6] **Mode** パラメータは **Equirectangular** に設定してください。

Miraikan

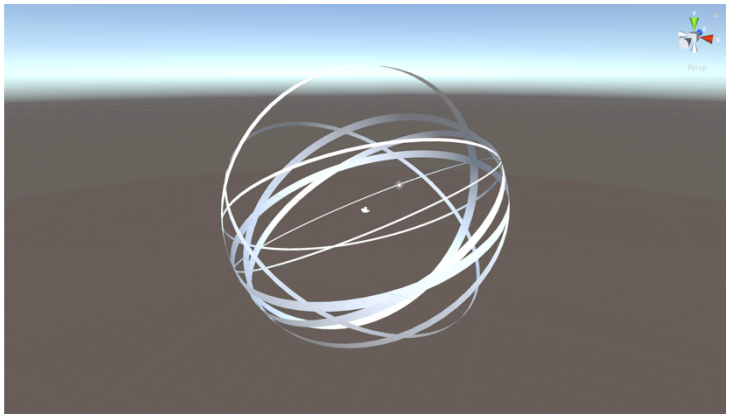
チュートリアル#4

レンダリング結果を良くする照明と各種設定値

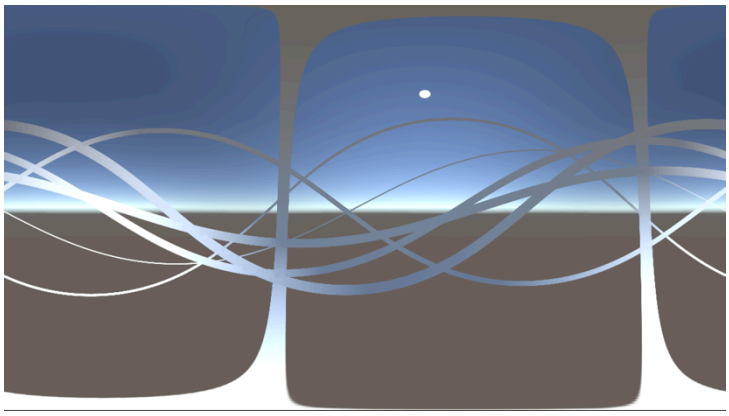
シーン「4. Lightings and Settings」を参照してください

チュートリアル＃４では、EquirectangularCamera を使ってジオ・コスモスのコンテンツを作る際のいくつかの設定について説明します。

用意したのは、いくつかのリングが回転している以下のようなシーンですが、作りたいのはフラットな白い線で描かれた輪が回っている映像です。



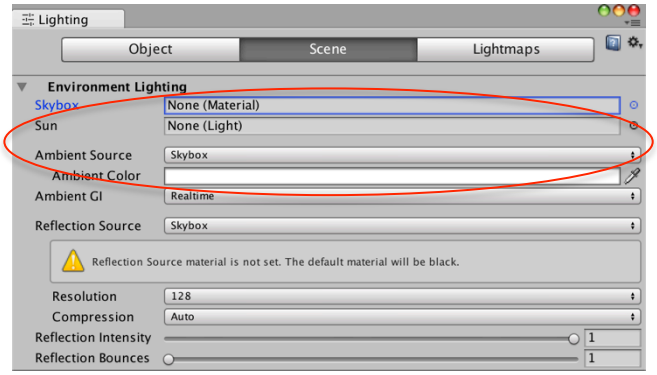
このシーンのレンダリング結果（ゲームビュー）を見ると以下のように見えます。



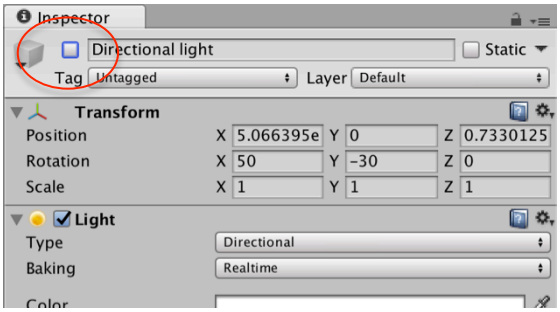
輪にグラデーションがかかっているように見えますが、これは照明の影響です。ジオ・コスモスの映像を作る際は、（特に意図して使う場合を除き）照明の影響を無くしておいた方が、混乱が少ないと思います。このシーンから照明の影響を取り除いてみましょう。

まず、環境光を整理します。

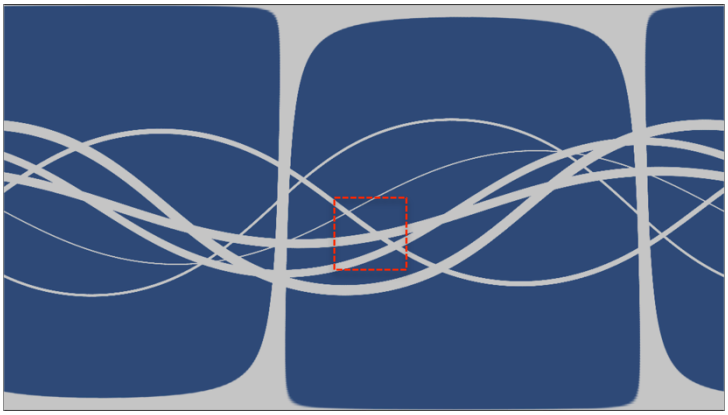
メニューの **Window > Lighting** から照明の設定画面を開きます。Scene タブでは編集中のシーンの環境光を設定できます。この中の **Skybox** を削除します。更に、**Ambient Source** を **Color** に、**Ambient Color** を白 100%に設定します。



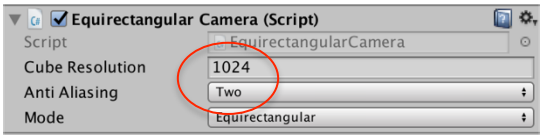
次に、シーンに配置されている **Directional Light** を OFF にします（ライトを削除するも可）。



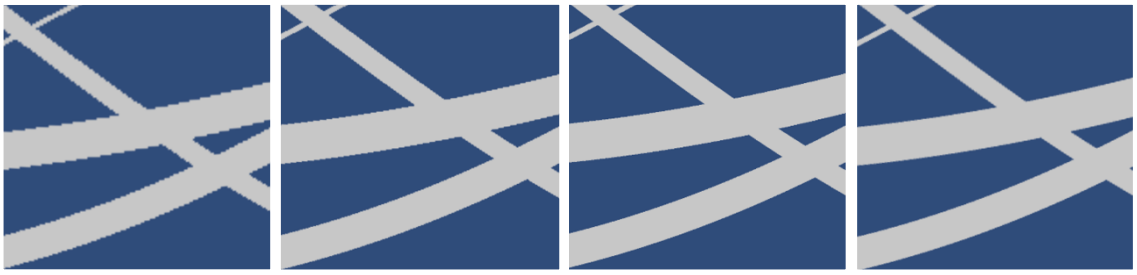
これで、ゲームビューに表示されるレンダリング結果がフラットな表現になりました。



次は、画質に関係する **EquirectangularCamera.cs** スクリプトの設定です。



以下の図は、ゲームビューの表示を 3840 x 2160 に設定した状態で、パラメータを変えて比較した様子です。（前ページの図で、赤点線で囲まれた部分の拡大）



解像度:512
AA : None

解像度:1024
AA : None

解像度:2048
AA : None

解像度:1024
AA : Two

4K の映像を生成するためには、2048 ピクセルの解像度があれば十分です。1024 の場合はアンチエイリアスをかけることで比較的良好な画質が得られる場合があります。画質への影響はコンテンツの性質（微細な表現要素の有無など）にも依りますので、処理負荷とのトレードオフで最適な設定値を決めてください。

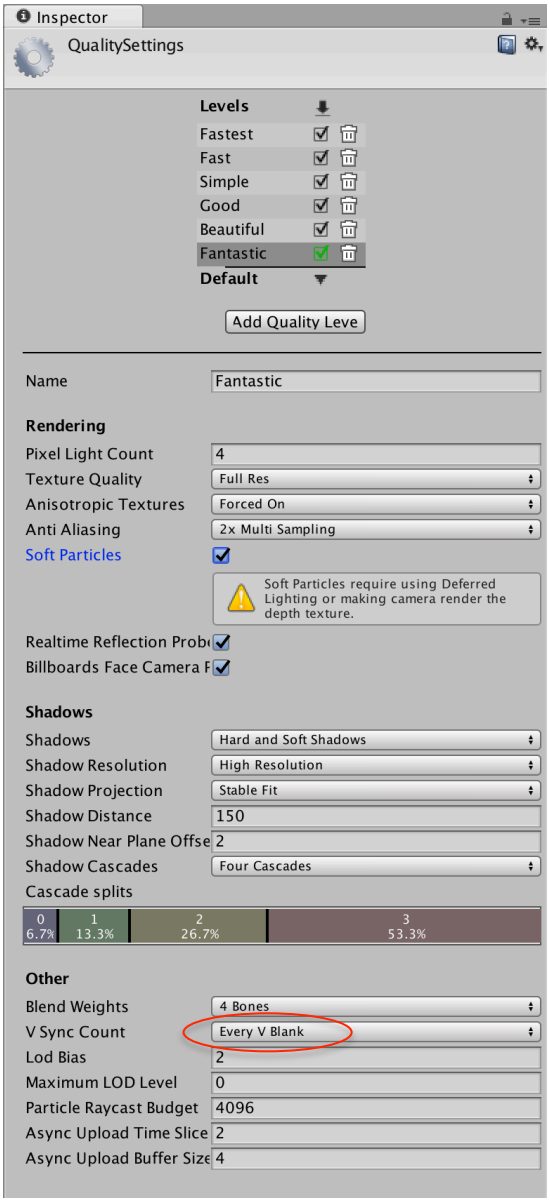
最後に、ジオ・コスモスのコンテンツ用にビルドする際に設定して頂きたい項目を挙げます。

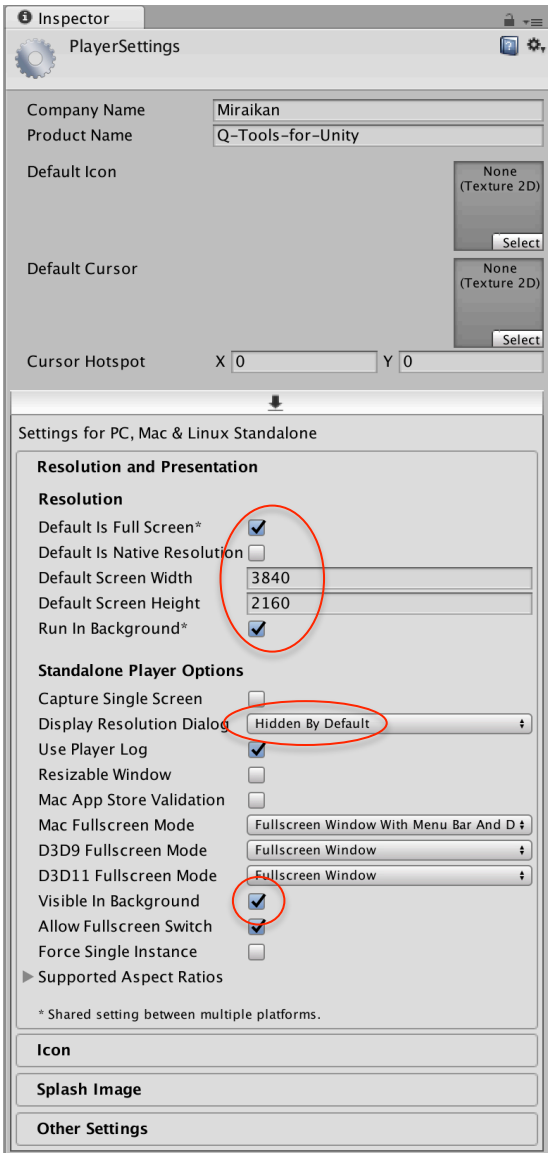
Edit > Project Settings > Quality Settings :

- **V Sync Count** は必ず **Every V Blank** に設定してください。

Edit > Project Settings > Player Settings :

- Resolution の各項目を次ページの図の値に設定してください。
- Background でも動作するように、**Run In Background** にチェックを入れてください。
- 必要な時に指定できるように、**Display Resolution Dialog** は **Hidden By Default** を指定してください。
- Background でも動作するように、**Visible In Background** にチェックを入れてください。


Inspector
QualitySettings
Levels
Fastest
Fast
Simple
Good
Beautiful
Fantastic
Default
Add Quality Leve
Name
Fantastic
Rendering
Pixel Light Count
4
Texture Quality
Full Res
Anisotropic Textures
Forced On
Anti Aliasing
2x Multi Sampling
Soft Particles
Soft Particles require using Deferred Lighting or making camera render the depth texture.
Realtime Reflection Prob
Billboards Face Camera F
Shadows
Shadows
Hard and Soft Shadows
Shadow Resolution
High Resolution
Shadow Projection
Stable Fit
Shadow Distance
150
Shadow Near Plane Offse
2
Shadow Cascades
Four Cascades
Cascade splits
0 6.7%
1 13.3%
2 26.7%
3 53.3%
Other
Blend Weights
4 Bones
V Sync Count
Every V Blank
Lod Bias
2
Maximum LOD Level
0
Particle Raycast Budget
4096
Async Upload Time Slice
2
Async Upload Buffer Size
4


Inspector
PlayerSettings
Company Name
Miraikan
Product Name
Q-Tools-for-Unity
Default Icon
None (Texture 2D)
Select
Default Cursor
None (Texture 2D)
Select
Cursor Hotspot
X 0
Y 0
Settings for PC, Mac & Linux Standalone
Resolution and Presentation
Resolution
Default Is Full Screen*
Default Is Native Resolution
Default Screen Width
3840
Default Screen Height
2160
Run In Background*
Standalone Player Options
Capture Single Screen
Display Resolution Dialog
Hidden By Default
Use Player Log
Resizable Window
Mac App Store Validation
Mac Fullscreen Mode
Fullscreen Window With Menu Bar And D
D3D9 Fullscreen Mode
Fullscreen Window
D3D11 Fullscreen Mode
Fullscreen Window
Visible In Background
Allow Fullscreen Switch
Force Single Instance
Supported Aspect Ratios
* Shared setting between multiple platforms.
Icon
Splash Image
Other Settings

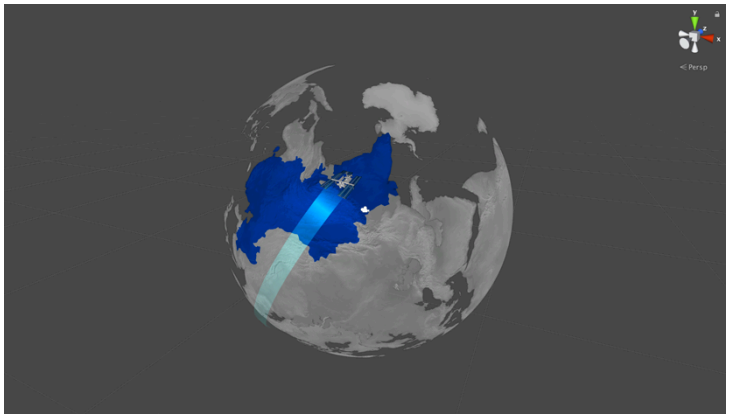
チュートリアル#5

注意を要するオブジェクトの前後関係と面の向き

シーン「5. Object Order and Polygon Face」を参照してください

最後に、少し実践的なコンテンツを例に、実際の作品を制作するときに注意しておきたいことを説明します。オブジェクトの前後関係と、面の向きについてです。

チュートリアル＃5 では、以下のようなシーンを用意しました。宇宙ステーションが地球を周回している様子です。



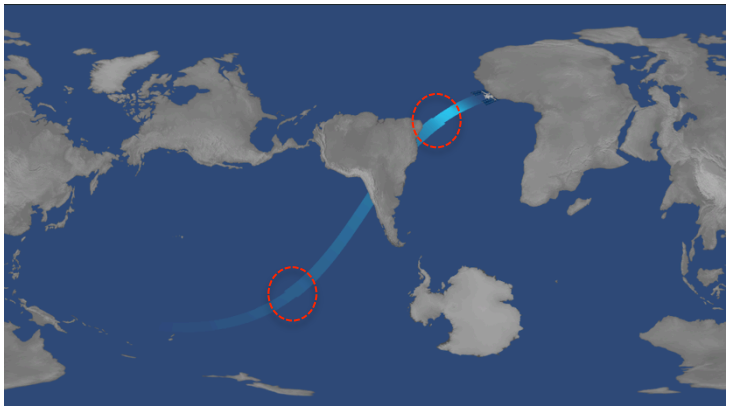
陸地は球面に沿ってモデリングされたメッシュオブジェクトで表現されています。海の部分には何もありません。宇宙ステーションは、球面に沿った円形（皿状）のメッシュに貼られたテクスチャーで表現されています。尾を引いている部分は、TrailRenderer を使ったエフェクトです。陸地の部分は、各国・地域ごとに別々のメッシュオブジェクトになっていて、宇宙ステーションが上空にあるときに個別に色を変えられるようになっています。（上空にあるかどうかの判断は Unity の当たり判定機能を使っています。）

メッシュには表裏があり、通常は表（おもて）面しかレンダリングされません。上の図のように、球体の外側から見て正しく見えるシーンは、中心に置かれたカメラから見ると面を裏から見ることになるため、レンダリングされないことになります。そこで、このシーンで使用している陸地メッシュや、テクスチャー用のメッシュは、**表裏両側に面を持つようにあらかじめ作られています**[7]。

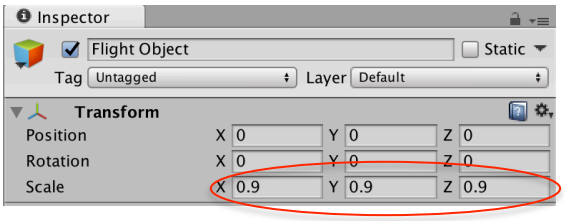
コンテンツを制作するときは、球体を外から見た視点で編集できるのが楽ですが、**カメラは内側から見ている**ことに留意してください。

[7] 裏面もレンダリングするようなマテリアルを使う、という方法もあります。

このシーンを再生した時のゲームビューは以下のように見えます。海の部分が青くなっているのは、Camera の設定で背景色を青に指定しているためです。



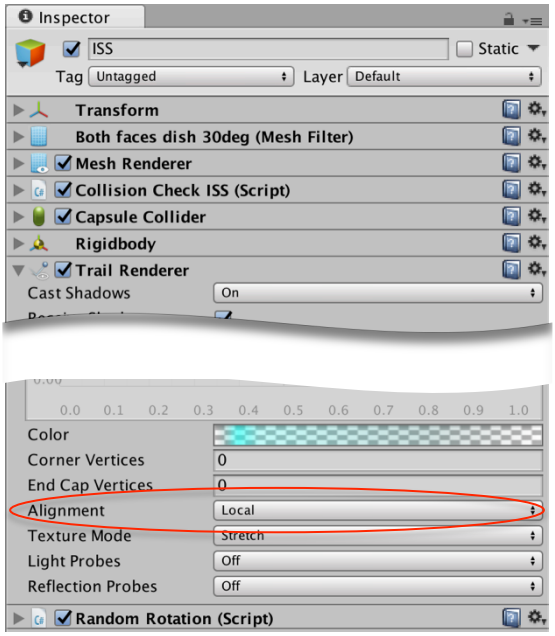
このシーンにはいくつか問題があります。まず、宇宙ステーションや尾を引いている青い線が、陸地の裏側に隠れています。これは、編集画面で正しく見えるように、宇宙ステーションや尾の部分を陸地より少し外側に配置しているためです。実際には、（カメラは内側から見ているので）陸地より内側に配置しなければなりません。宇宙ステーションが含まれた **Flight Object** という名前のオブジェクトのサイズを、1.1 から 0.9 に変えてみましょう。



これで、陸地より手前に表示されるようになりました。

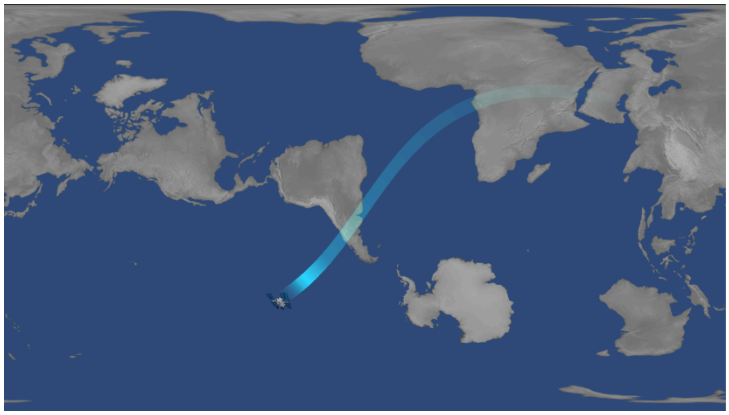
もう一つの問題は、やや分かりにくいですが、尾の部分の太さが変わっている箇所があります（図の赤点線で示した部分）。これは、6 つのカメラで撮影した画像を繋いでいることと関係があります。尾の表現に使っている TrailRenderer やパーティクルエフェクト、地形で使われるビルボードなど、Unity のいくつかの要素には、面の向きがカメラと正対するように自動的に最適化されるものがあります。そのため、異なるカメラで撮影された画像は、それぞれ面の向きが異なってしまい正しく繋がりません。ジオ・コスモス用のコンテンツでこうした要素を使用する場合は、面をカメラに向ける機能を OFF にする方が良いです。

例えば、TrailRenderer の場合は、**Alignment** を **View** から **Local** に変えることで、この機能を OFF にすることができます[8]。



ISS オブジェクトのインスペクタ

これらを修正すると、ゲームビューでの見え方は以下のように期待通りのものになりました。



[8] Unity5.5 以降の機能です。

Miraikan

チュートリアル#6

オリジナルのシーンを作ってみる

シーン「6. Try to build your Scene」を参照してください

ここまでのチュートリアルで、Unity を使ってジオ・コスモスのコンテンツを制作する方法の紹介は一通り終わりました。最後のチュートリアルでは、皆さんのオリジナルのシーンを作り、動かしてみましょう。

チュートリアル＃6 に用意したシーンは、Unity で新規シーンを作成した直後の、Camera オブジェクトと Light オブジェクトだけが置かれている状態です。
この Camera オブジェクトに **EquirectangularCamera.cs** スクリプトをアタッチするか、Camera オブジェクトを削除して **EquirectangularCamera** プレファブで置き換えてください。
スクリプトもプレファブも、**/Q-Tools/EquirectangularCamera/** の下に入っています。

