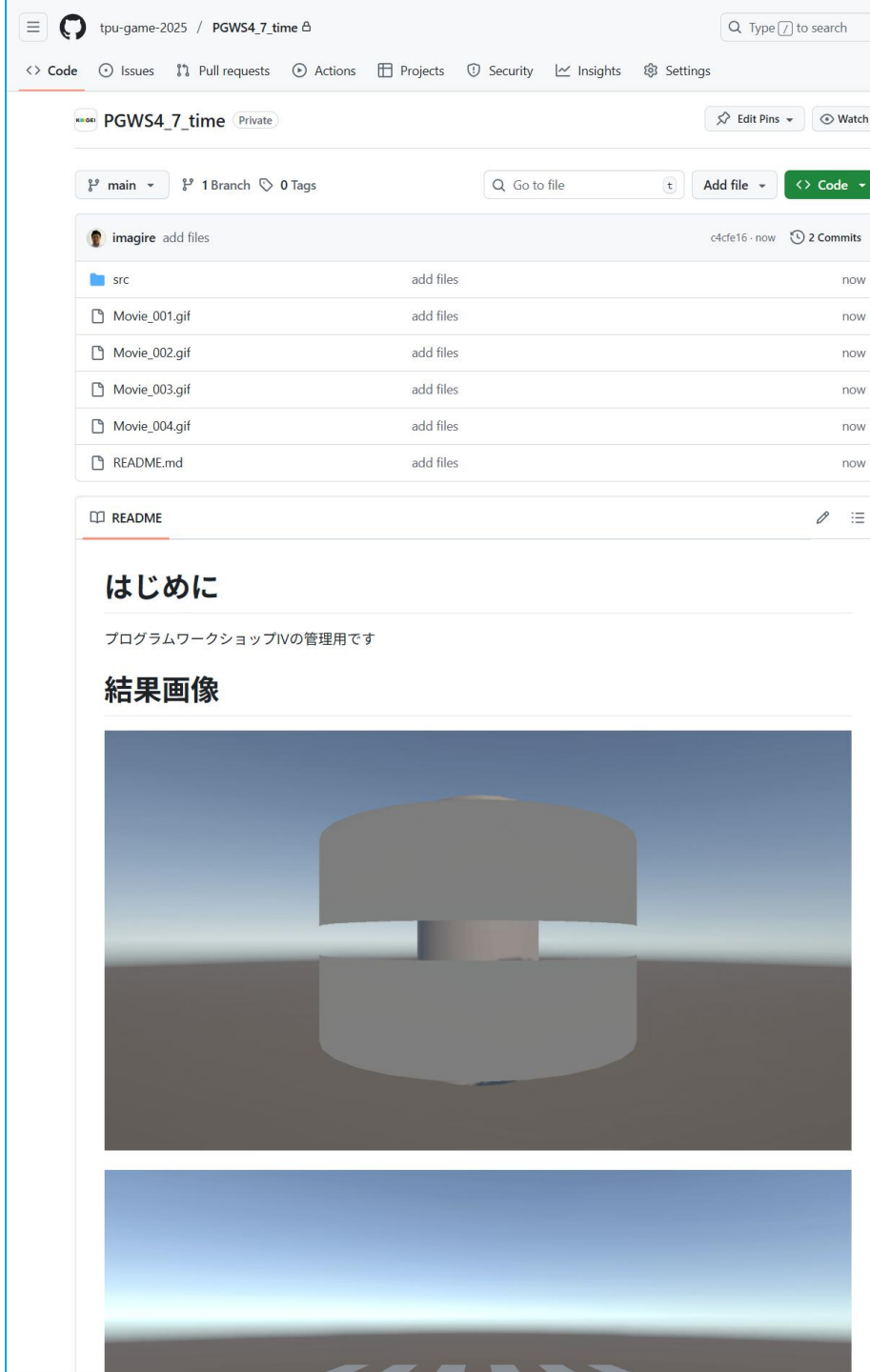
時間変化

2025年度 プログラム ワークショップIV（7）

本日の課題

- https://github.com/tpu-game-2025/PGWS4_7_time
  をforkして、「結果画像」に結果を貼ってください
  - 今回の範囲で何か試してください
    - 「Scene_4_MyBest」に



本日の内容

- 光を走らせる
  - 概要
  - 回してみよう
  - 光を走らせてみよう
  - 思った場所に走らせよう

本日の内容

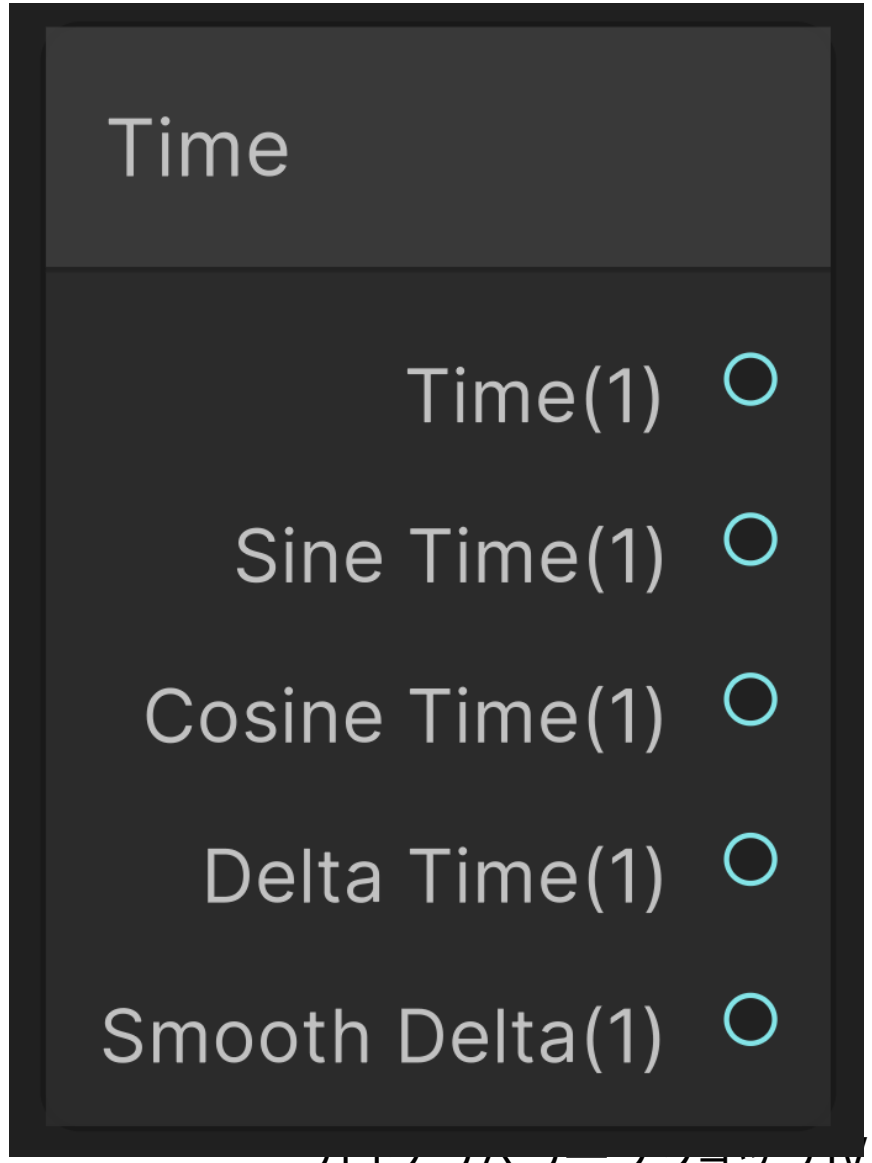
- 光を走らせる
  - 概要
  - 回してみよう
  - 光を走らせてみよう
  - 思った場所に走らせよう

今日の話

- 光を流してみよう

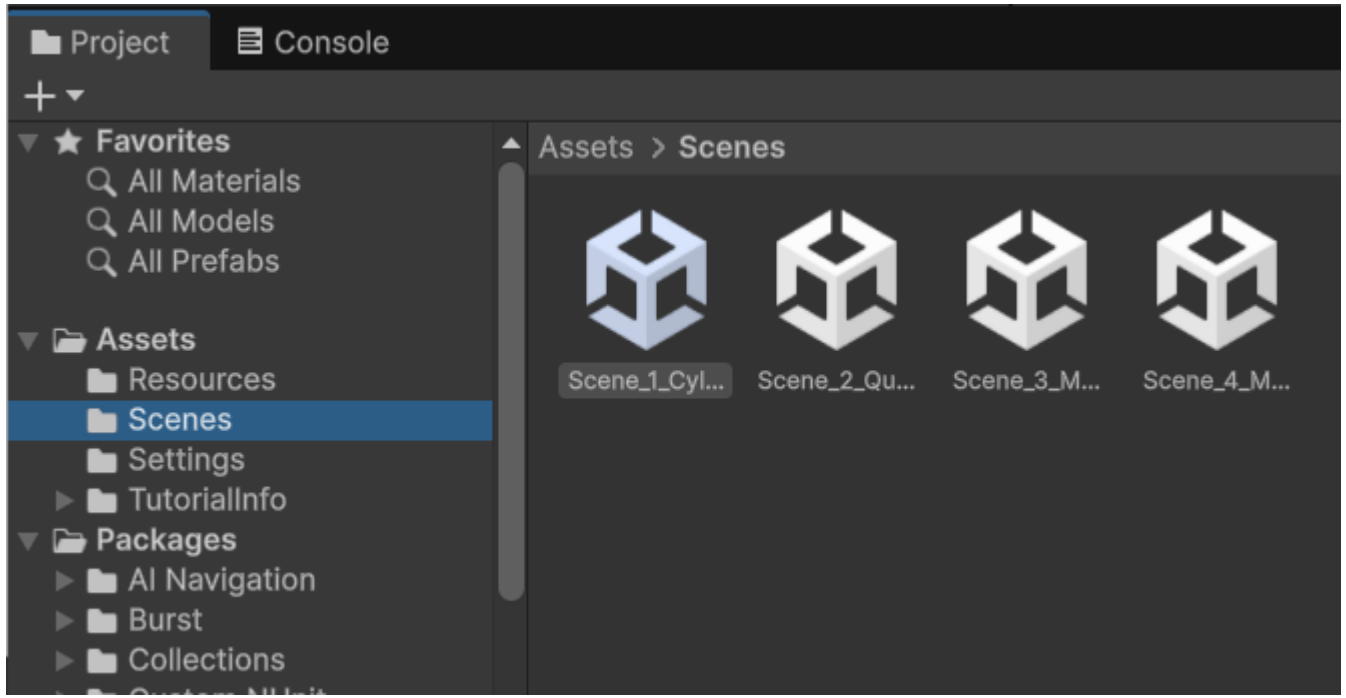
今日の主役:タイムノード

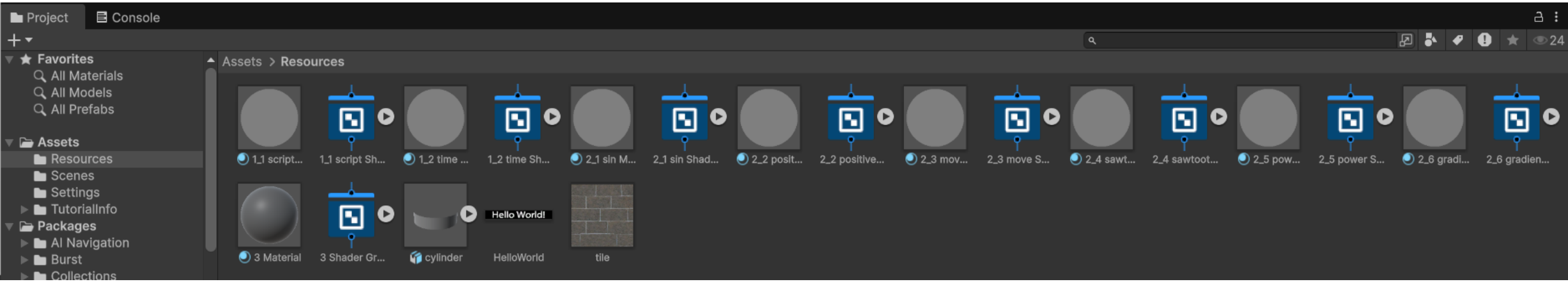
Name	Direction	タイプ	バインディング	説明
Time	出力	Vector 1	なし	Time の値
Sine Time	出力	Vector 1	なし	Time 値の正弦
Cosine Time	出力	Vector 1	なし	Time 値の余弦
Delta Time	出力	Vector 1	なし	現在のフレーム時間
Smooth Delta	出力	Vector 1	なし	平滑化された現在のフレーム時間



アセット

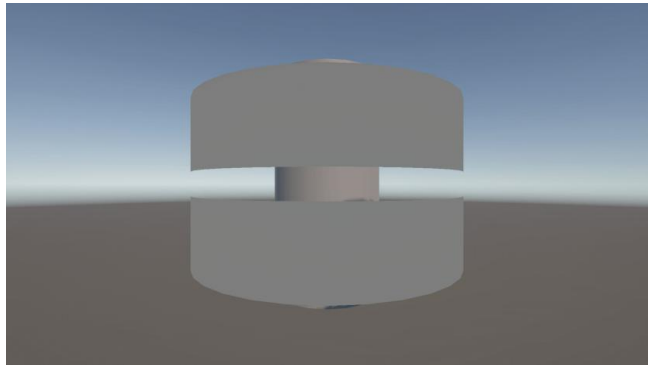
- 4シーン
- 1モデル(FBX)
- 2テクスチャ(透過PNG)
- 9マテリアル＆Shader Graph

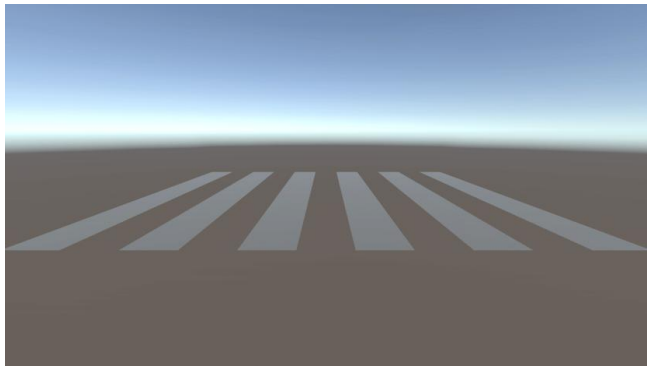


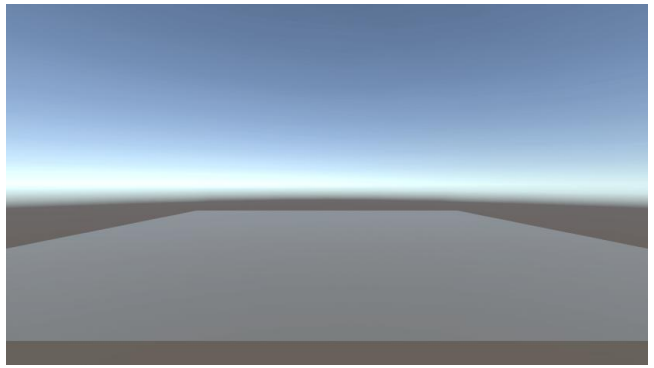


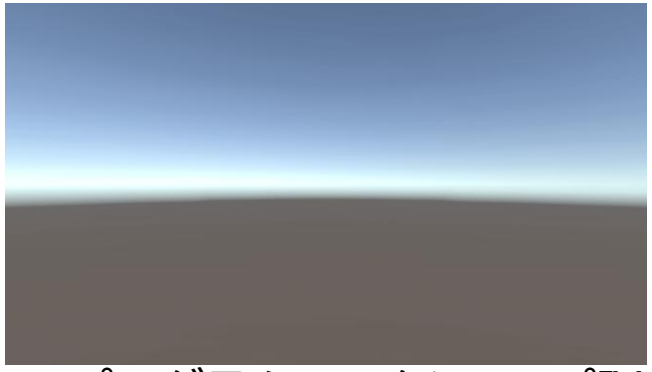
シーン

- Scene_1_Cylinder
  - 円柱の周りをスクリプトとシェーダでHello World!を回してください
- Scene_2_Quad
  - ステップを踏んで、最終的に光の筋を作ってください
- Scene_3_Mask
  - 部分的に光を走らせてください
- Scene_4_MyBest
  - 最高のシーンを作ってください

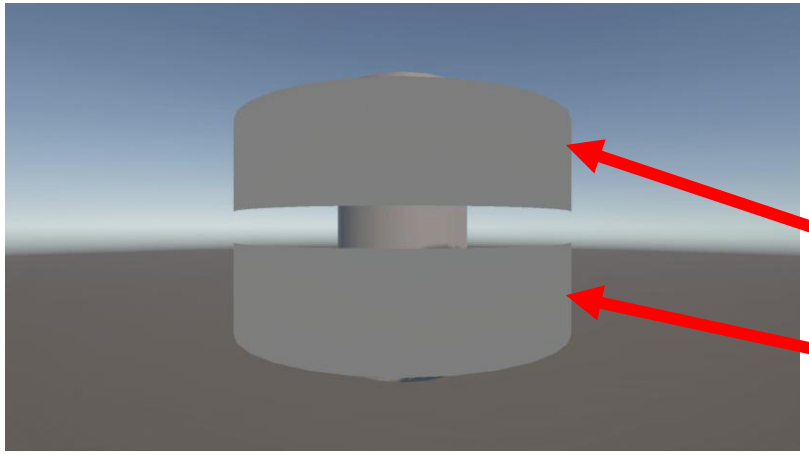




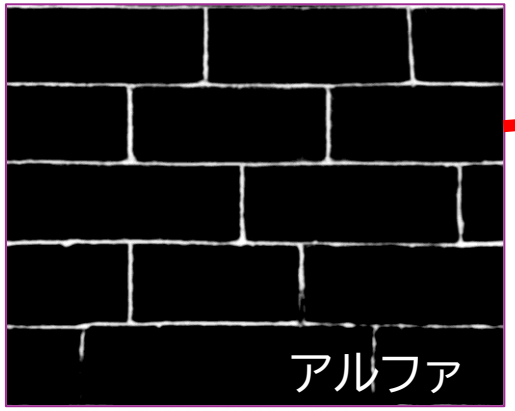


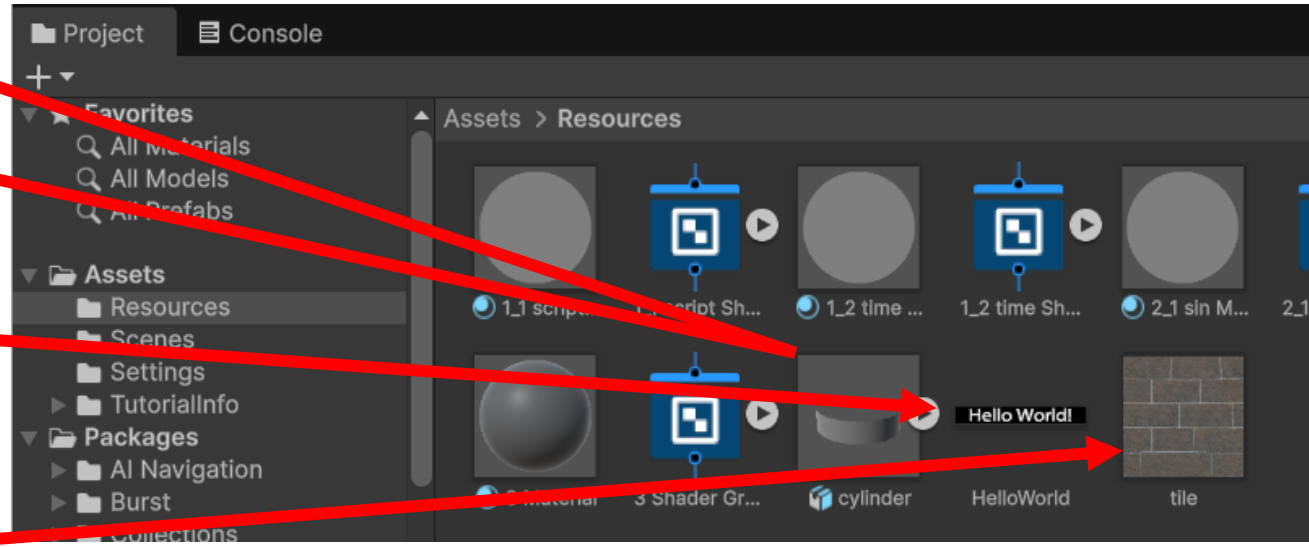


モデル・テクスチャ

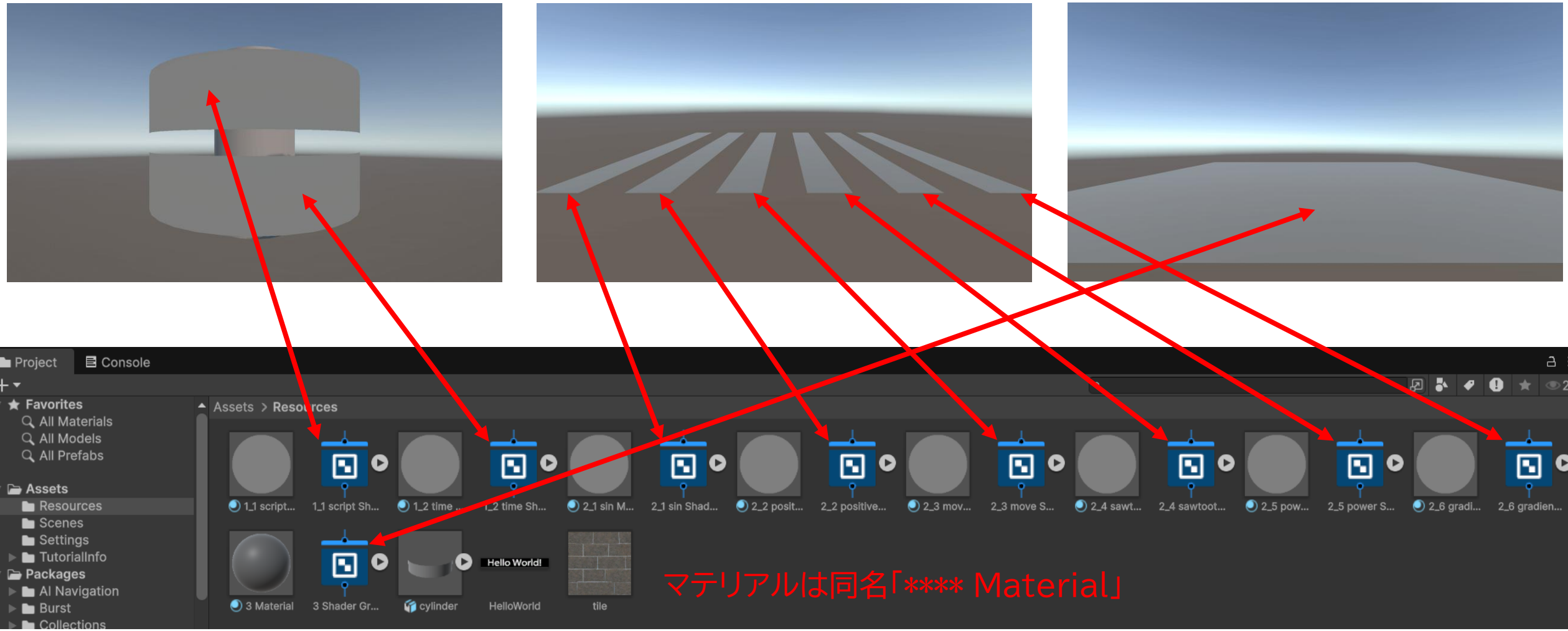


Hello World!





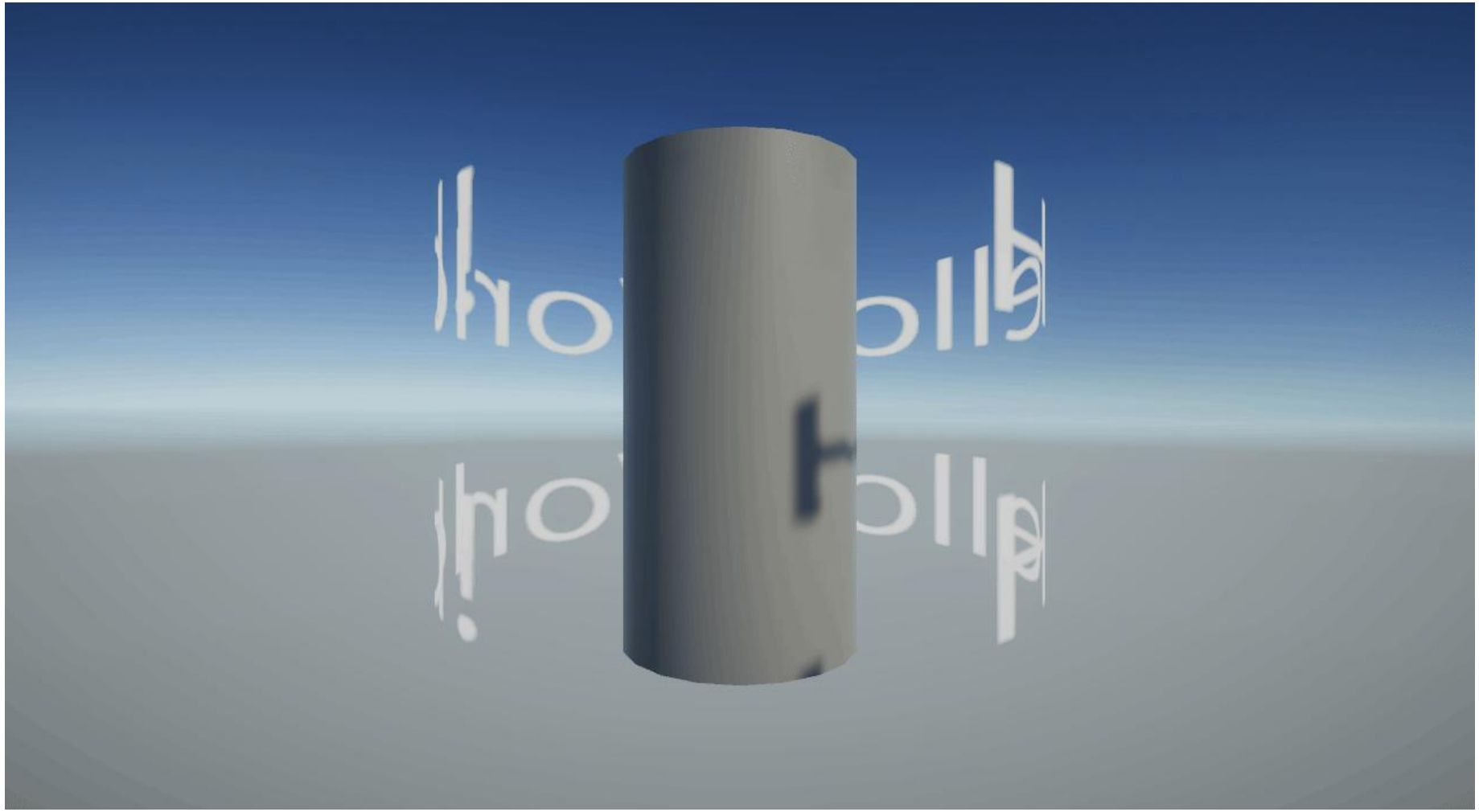
マテリアル＆Shader Graph



本日の内容

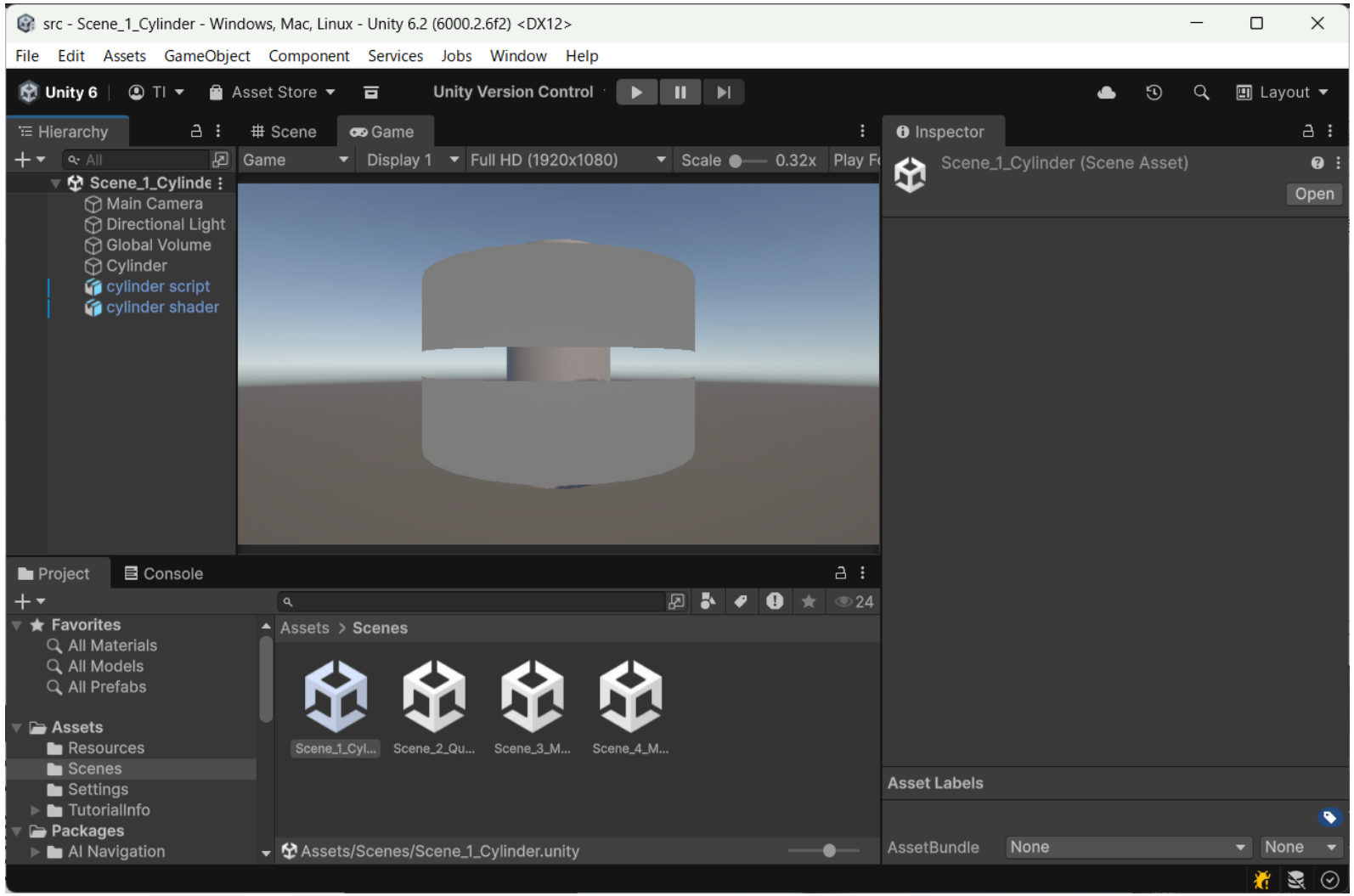
- 光を走らせる
  - 概要
  - 回してみよう
  - 光を走らせてみよう
  - 思った場所に走らせよう

回してみよう



シーン

- Scene_1_Cylinder

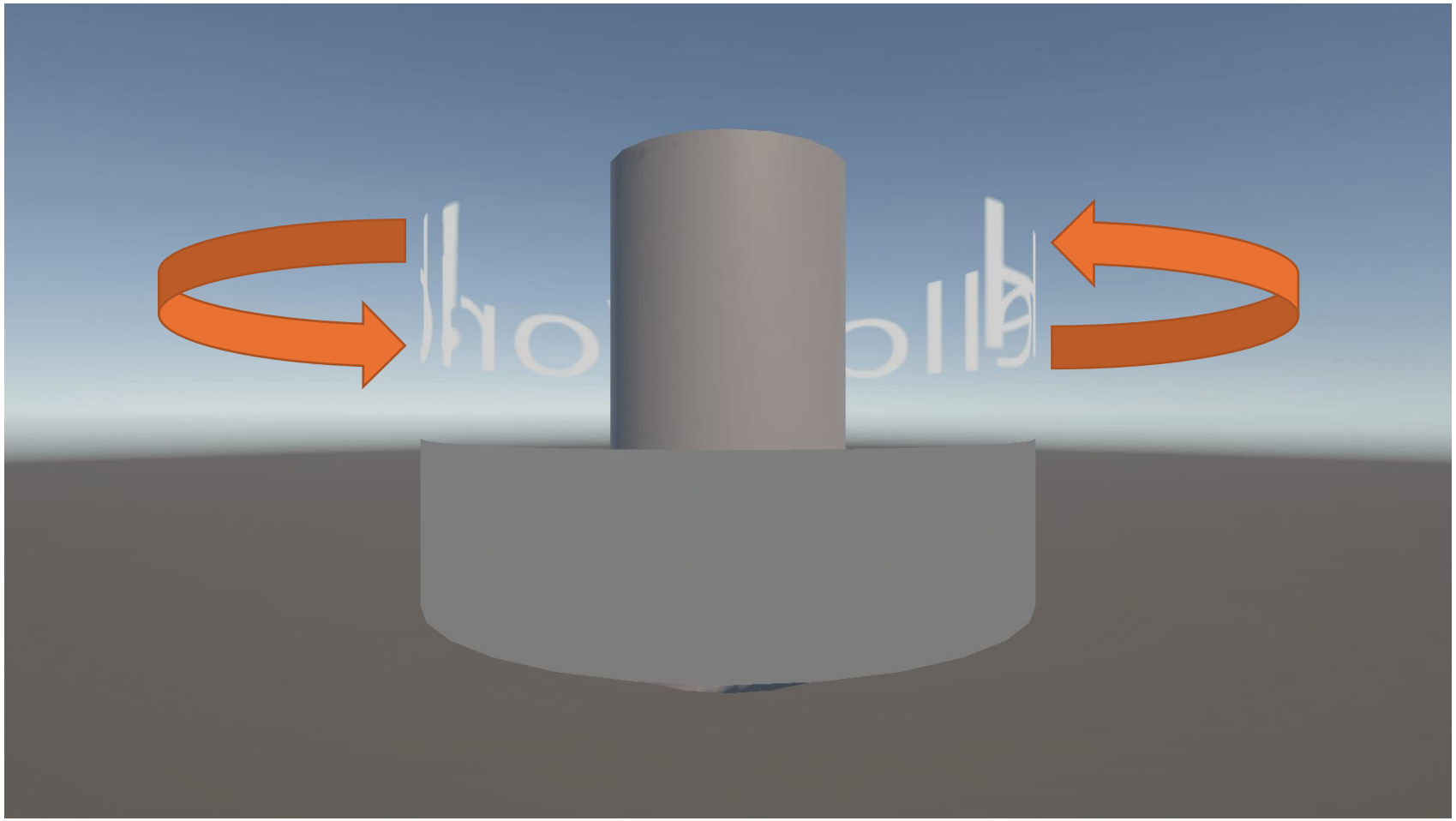


本日の内容

- 光を走らせる
  - 概要
  - 回してみよう
    - オブジェクトを動かす
  - 光を走らせてみよう
  - 思った場所に走らせよう

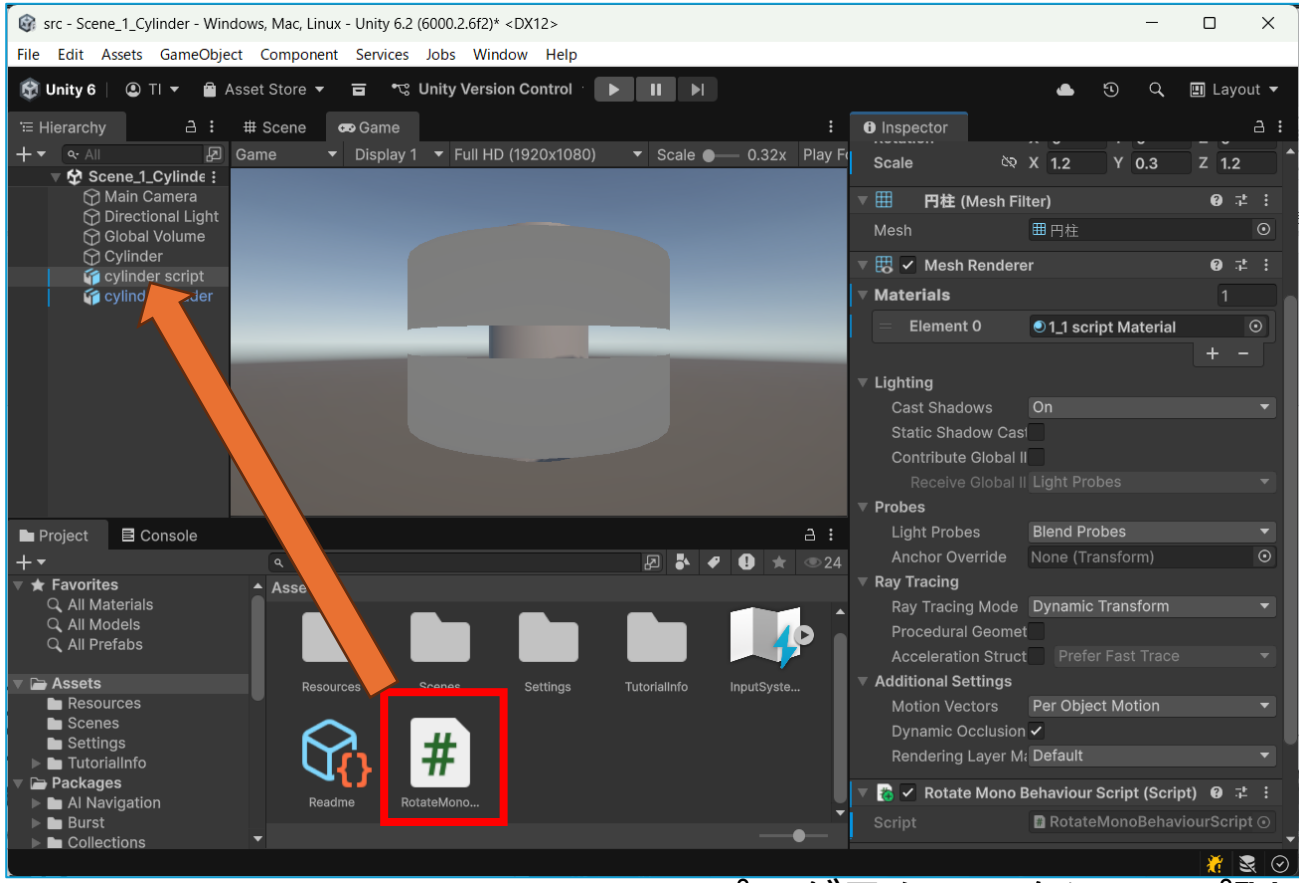
オブジェクトを動かす

- スクリプトで回転



スクリプトの設定

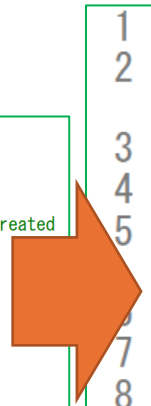
- Projectウィンドウで
  右クリック
  - 「Create」で
    「MonoBehavior Script」を生成
- イイ感じにRename
  - RotateMonoBehaviourScript
    とか
- 動かしたいオブジェクトにD&D
  - 配置済みの物体
    「cylinder script」



スクリプト

- transformをRotateで回す
  - 一周 360.0f
  - 各フレームで回す角度を指定
    - 角度の直接指定ではない

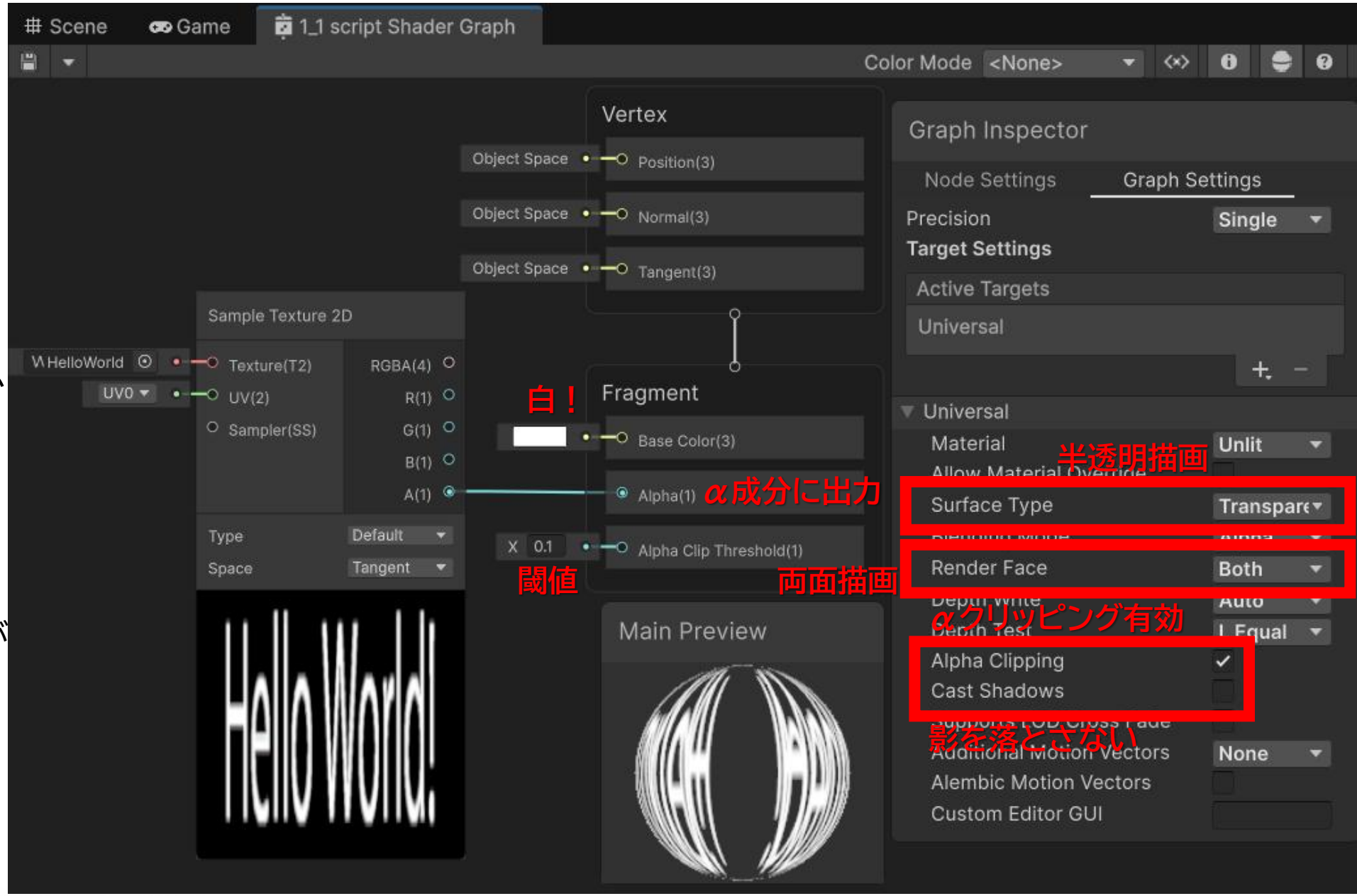
```
1       using UnityEngine;
2
3       public class RotateMonoBehaviourScript : MonoBehaviour
4       {
5           // Start is called once before the first execution of Update after the MonoBehaviour is created
6           void Start()
7           {
8
9           }
10
11          // Update is called once per frame
12          void Update()
13          {
14
15          }
16      }
```



```
1       using UnityEngine;
2
        Unity スクリプト | 0 個の参照
3       public class RotateMonoBehaviourScript : MonoBehaviour
4       {
5           // Update is called once per frame
            Unity メッセージ | 0 個の参照
6           void Update()
7           {
8               //                  360で一周　1秒で1増える　1周の秒数
9               float rotationSpeed = 360.0f * Time.deltaTime / 10.0f;
10              this.transform.Rotate(0, rotationSpeed, 0);
11          }
12      }
```

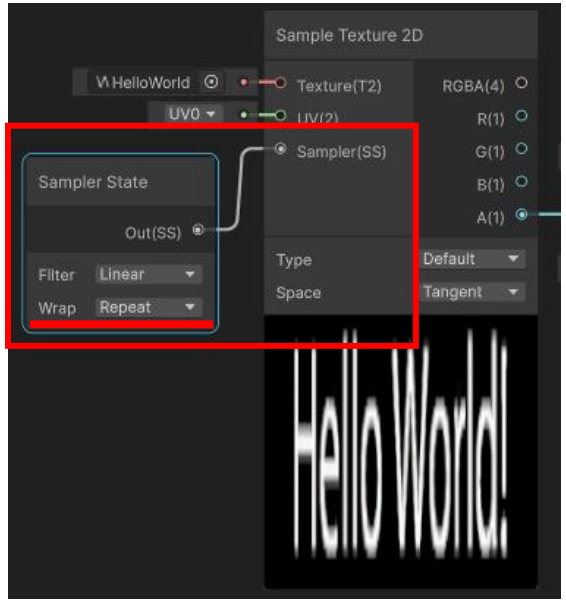
シェーダ

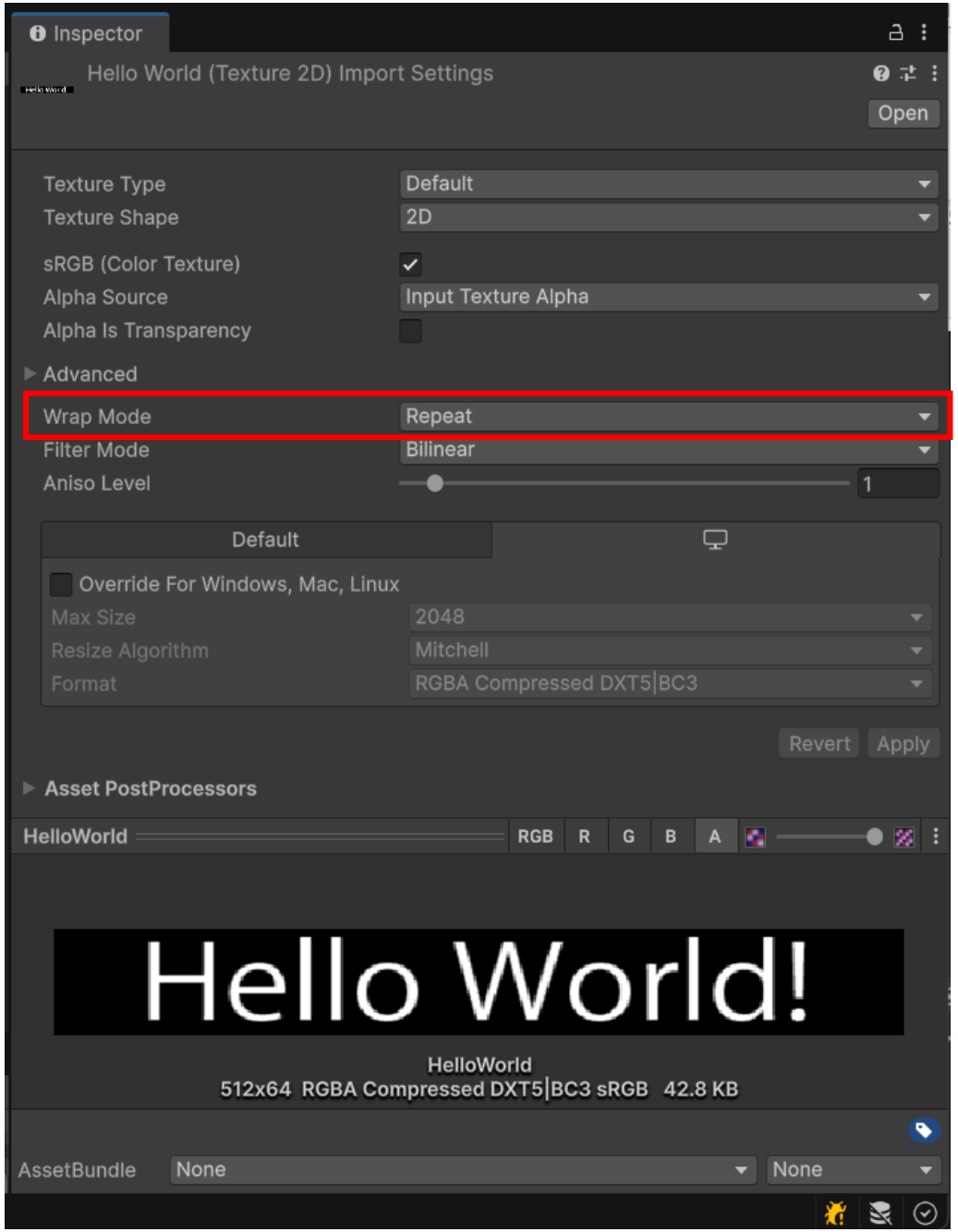
- 半透明描画
- α成分に出力
- 両面描画
- αクリッピング
  - α値が閾値よりも小さいと、描画されない
    - フレームバッファとの合成が減るので、ちょっと軽くなる
- 影を落とさない



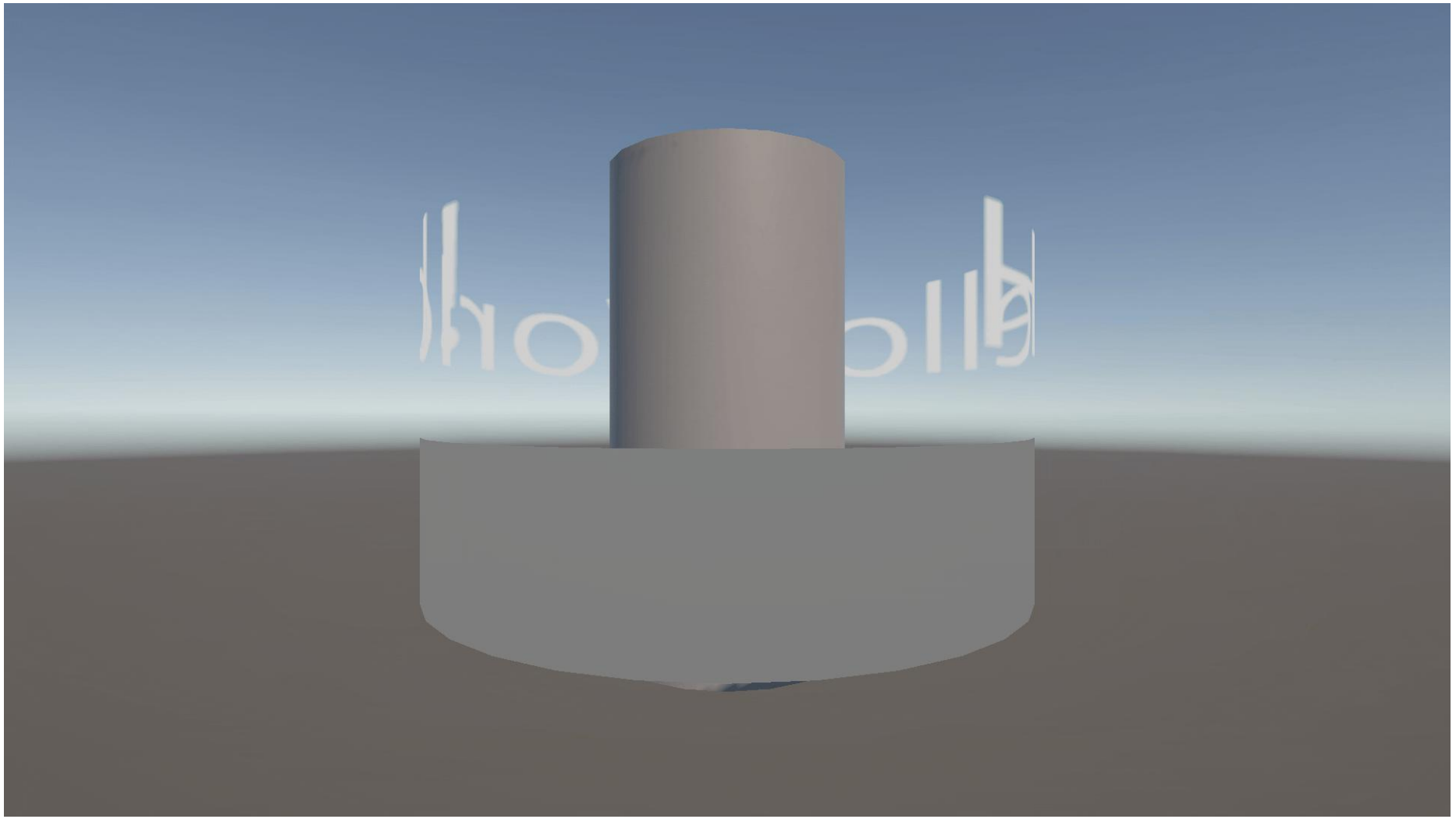
注意:サンプラー設定

- テクスチャの繰り返し(Wrap Mode)は、「Repeat」に設定
  - 今回は設定済み
- 強制的に設定するにはサンプラー状態のノードを追加する





やってみよう



本日の内容

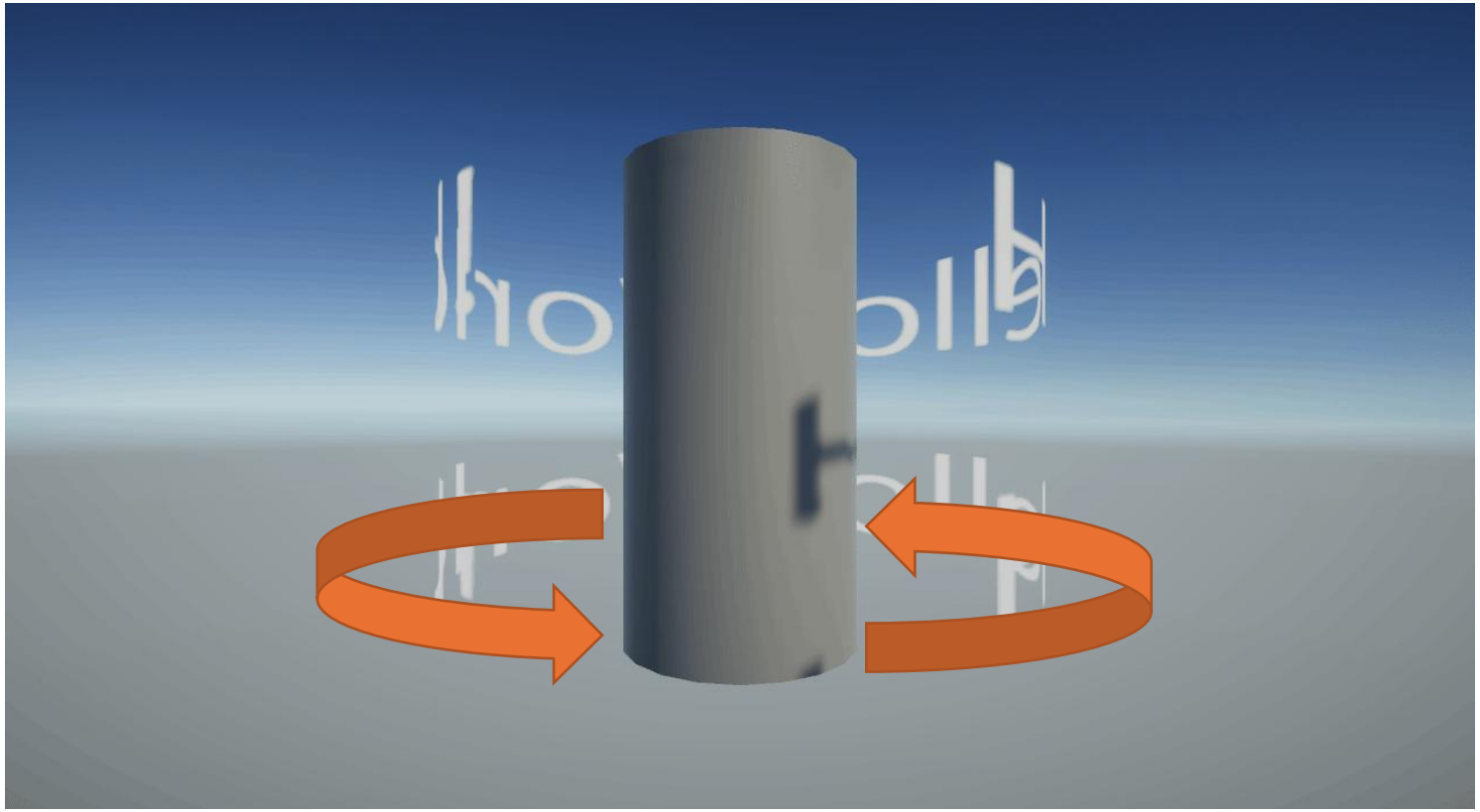
- 光を走らせる
  - 概要
  - 回してみよう
    - オブジェクトを動かす
    - テクスチャをスクロールさせる
  - 光を走らせてみよう
  - 思った場所に走らせよう

どうやって回す？

- ~~オブジェクトを動かす~~
  - ~~スクリプトで回転~~

この授業の狙いではない

- シェーダだと何がいいの？
  - CPU負荷が下がる
  - 描画と挙動の分業化
    - スクリプトはもっと難しい処理に専念できる
    - それぞれが独立して品質を突き詰められる
- どうすればシェーダで同じことを実現できるか？

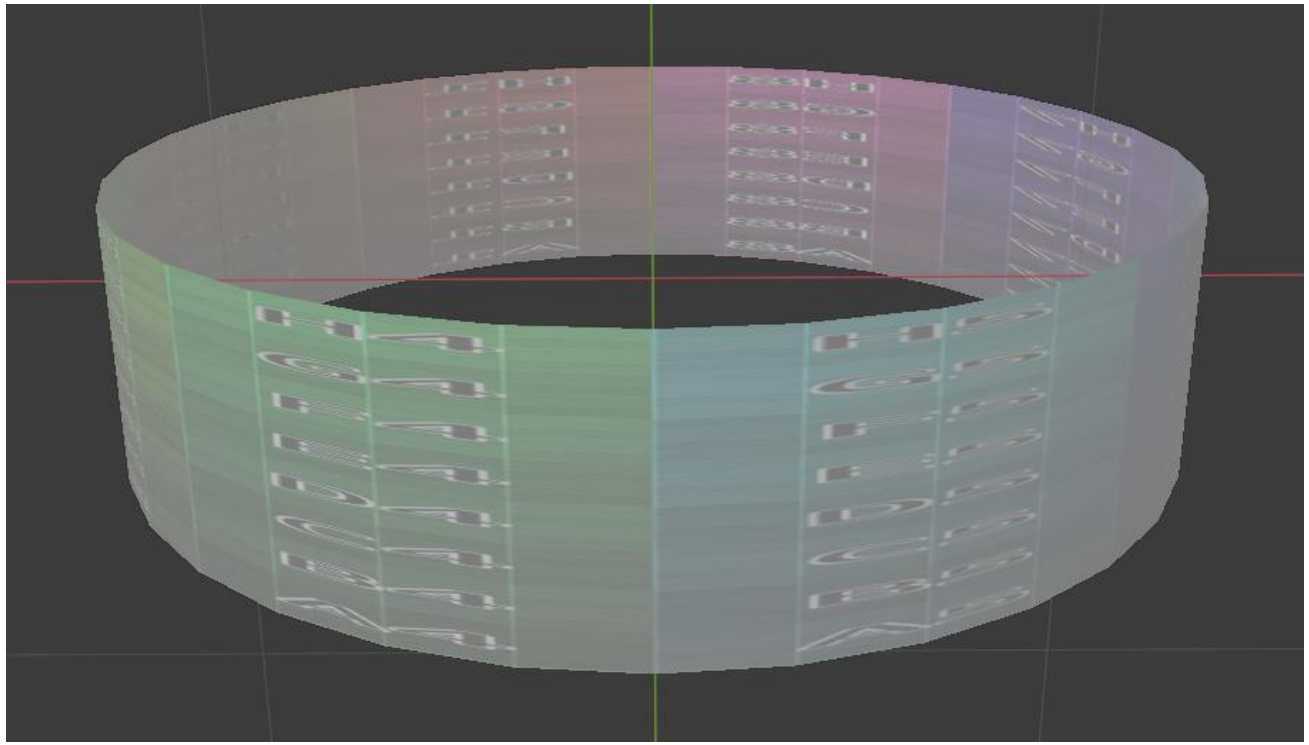


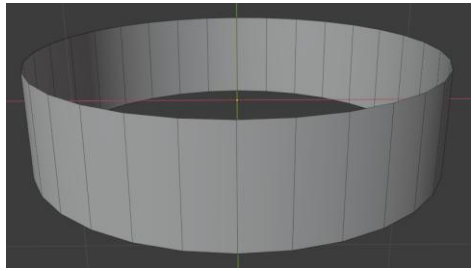
発想の逆転

- 物体を回す
- テクスチャを読み込む位置を変える
  - 物体は固定
  - 「UVスクロール」といわれることが多い

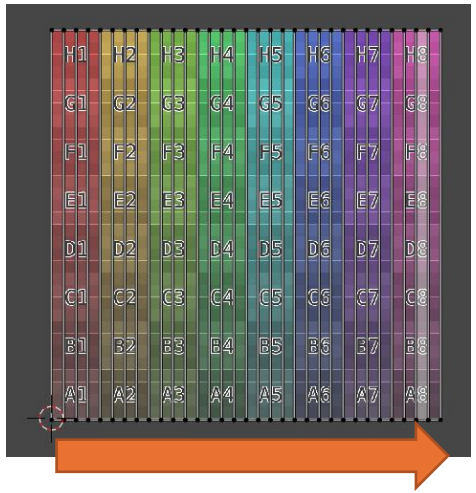
用意したモデル

cylinder.fbx





メッシュ



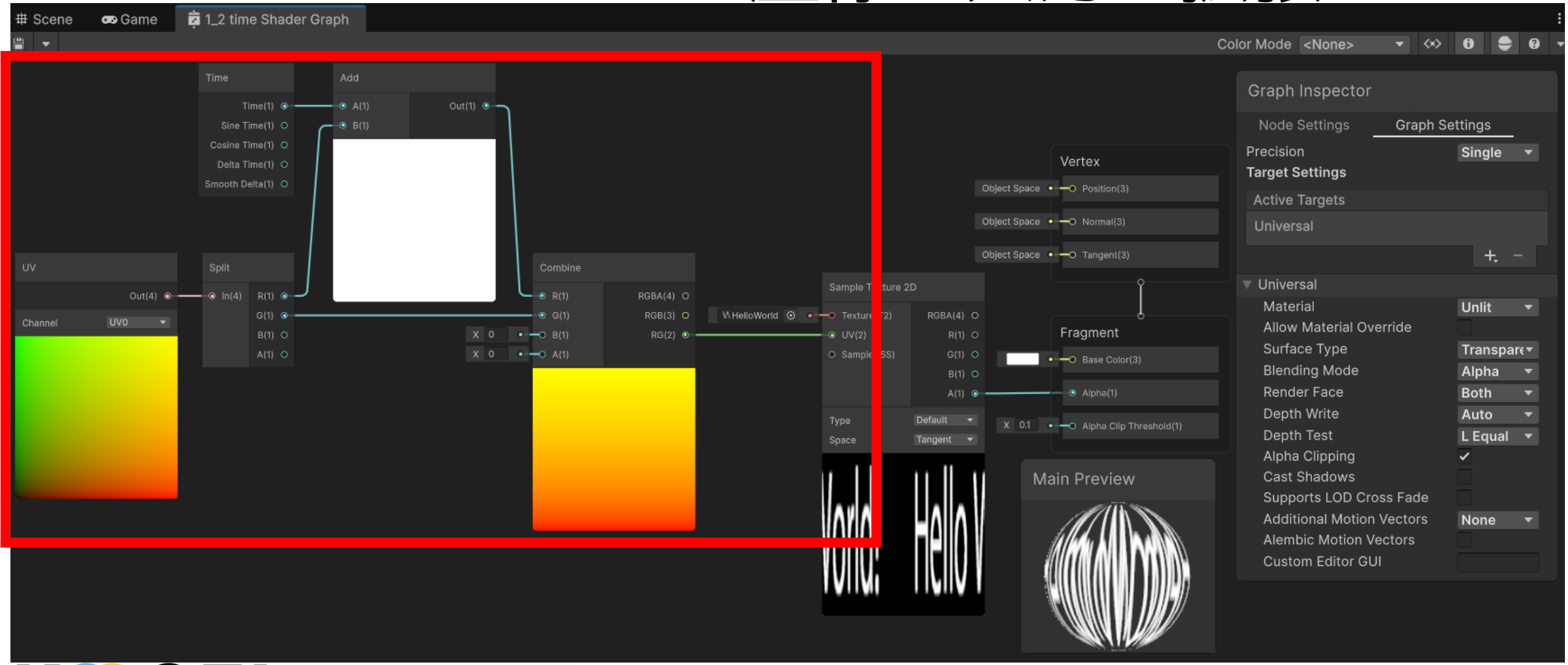
UV

U方向にテクスチャを参照する位置を
変えると流れるように表示される

HelloWorld.png

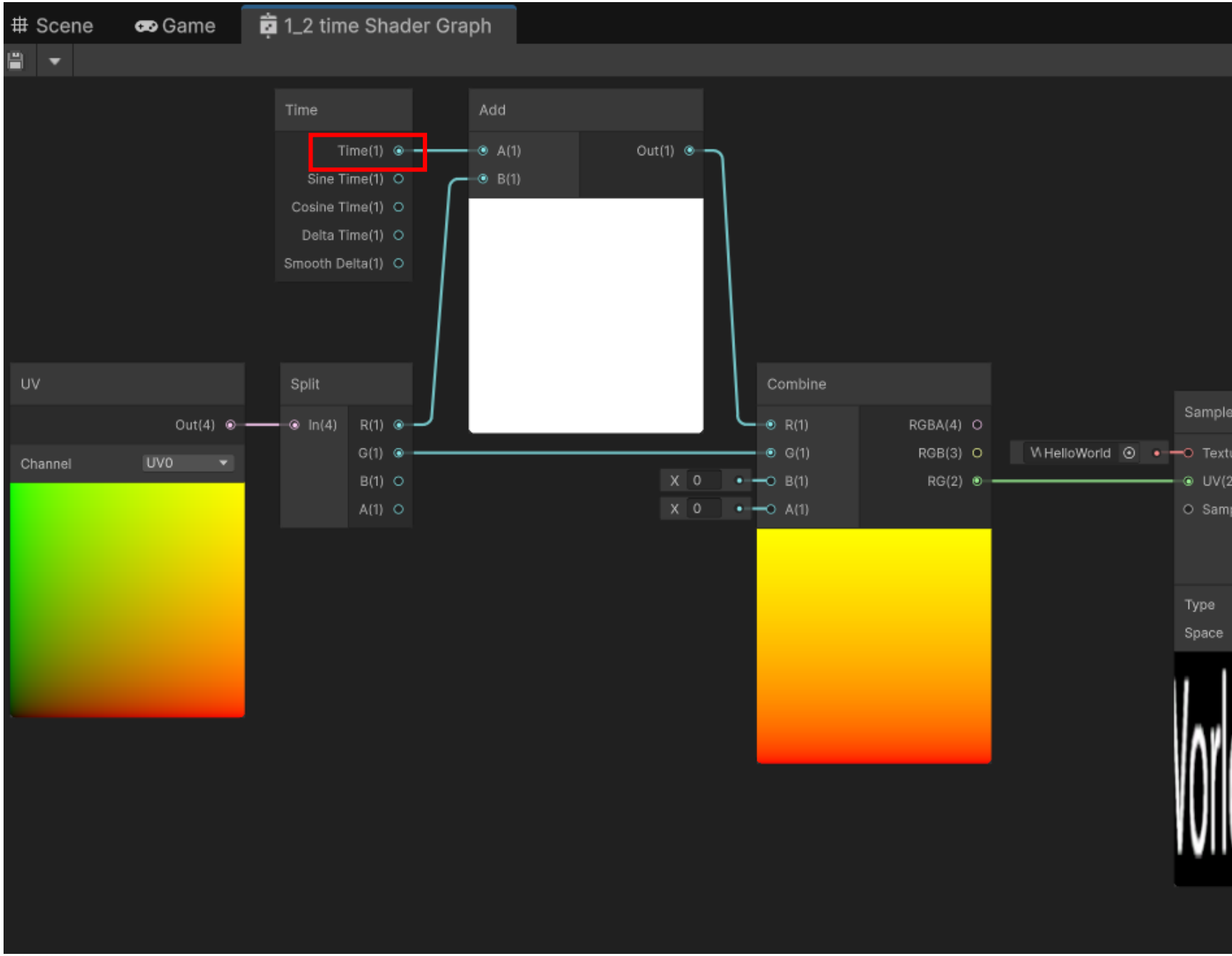
RGBAですべて同じ値

シェーダ：テクスチャ座標の入力を拡張



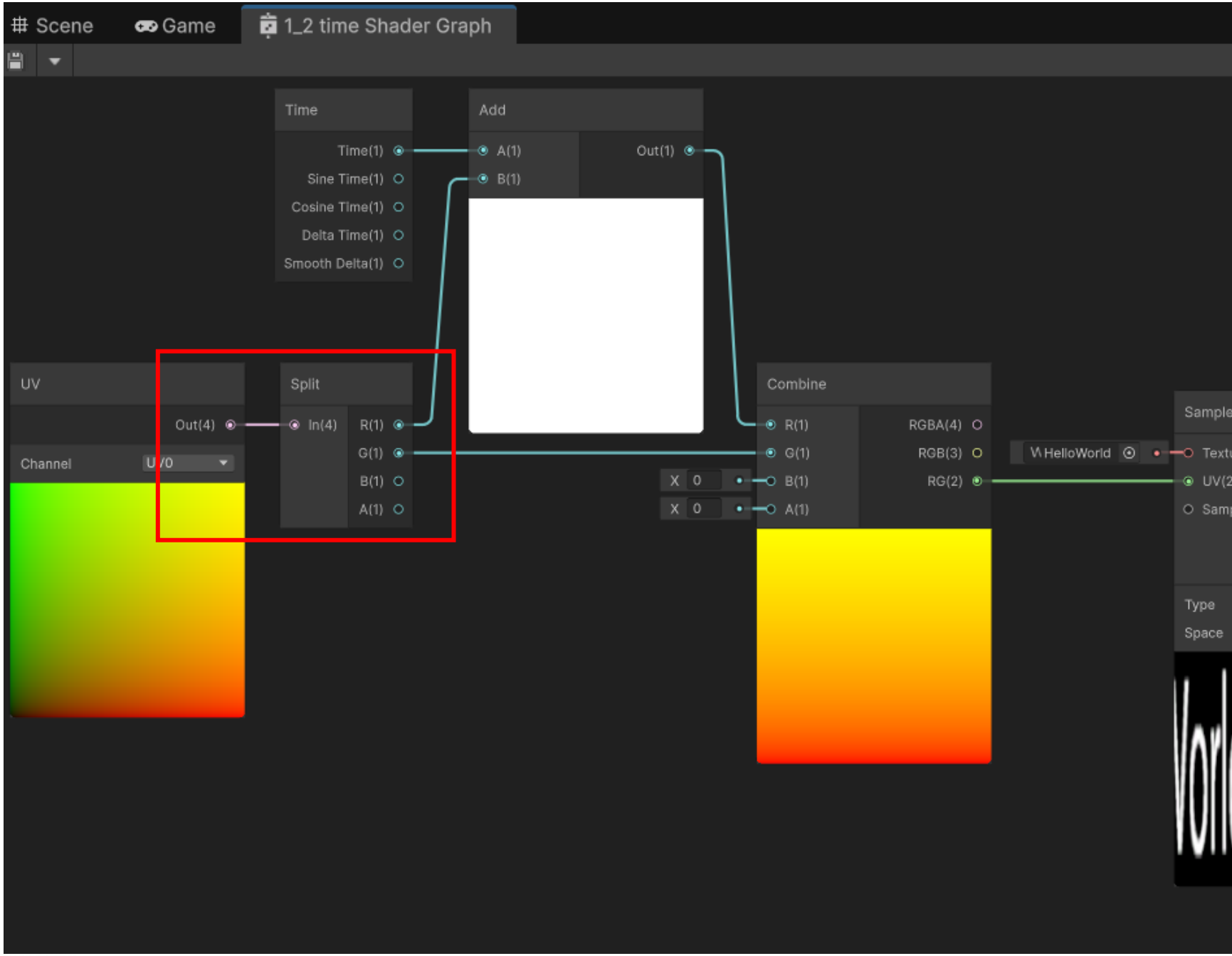
シェーダで回す

- 時間を取っくる
  - Timeノード
  - Time出力
    - 1秒で1.0増える



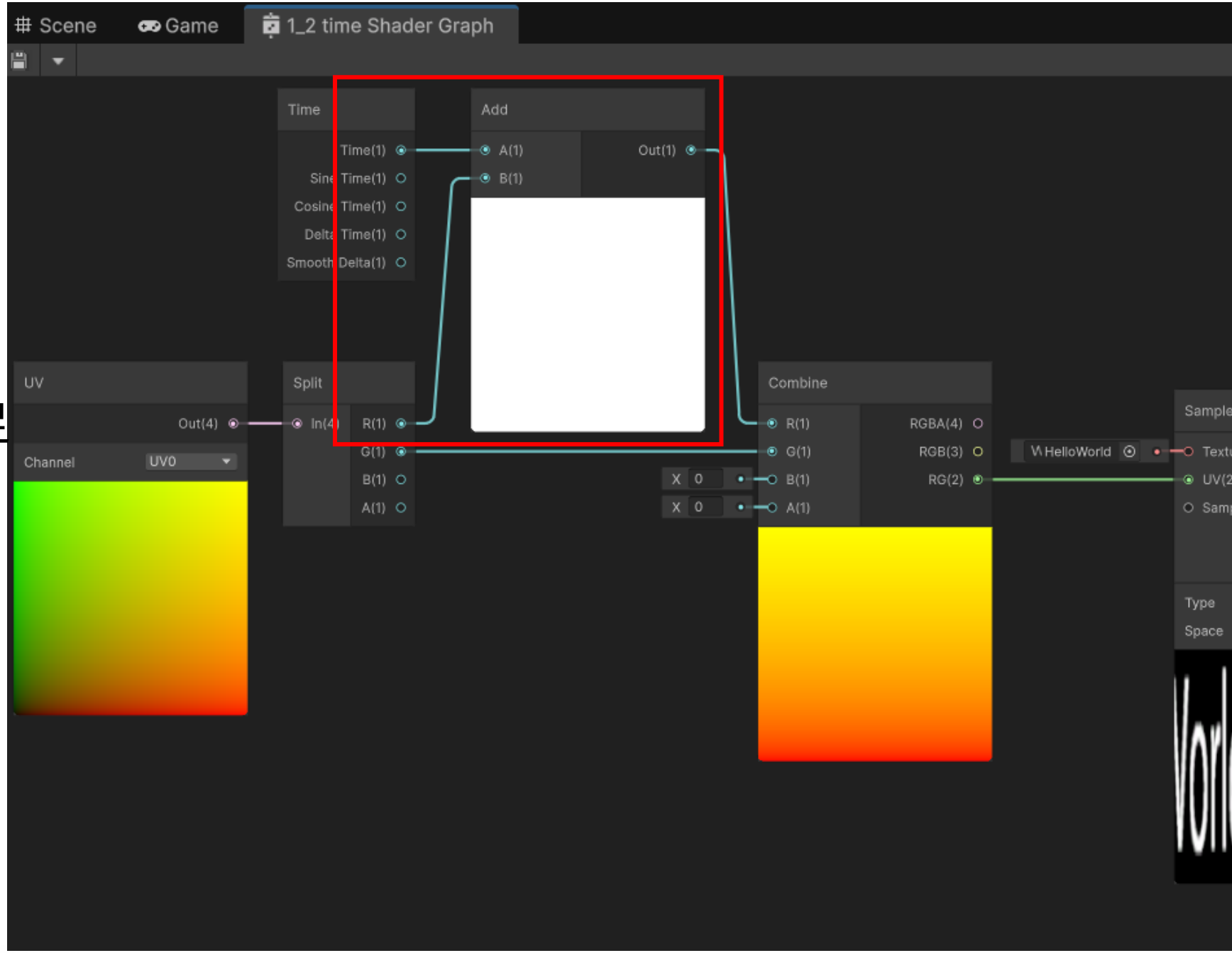
シェーダで回す

- 横方向のサンプリング位置を取得
  - UVのU成分
  - Splitノードで成分を分離



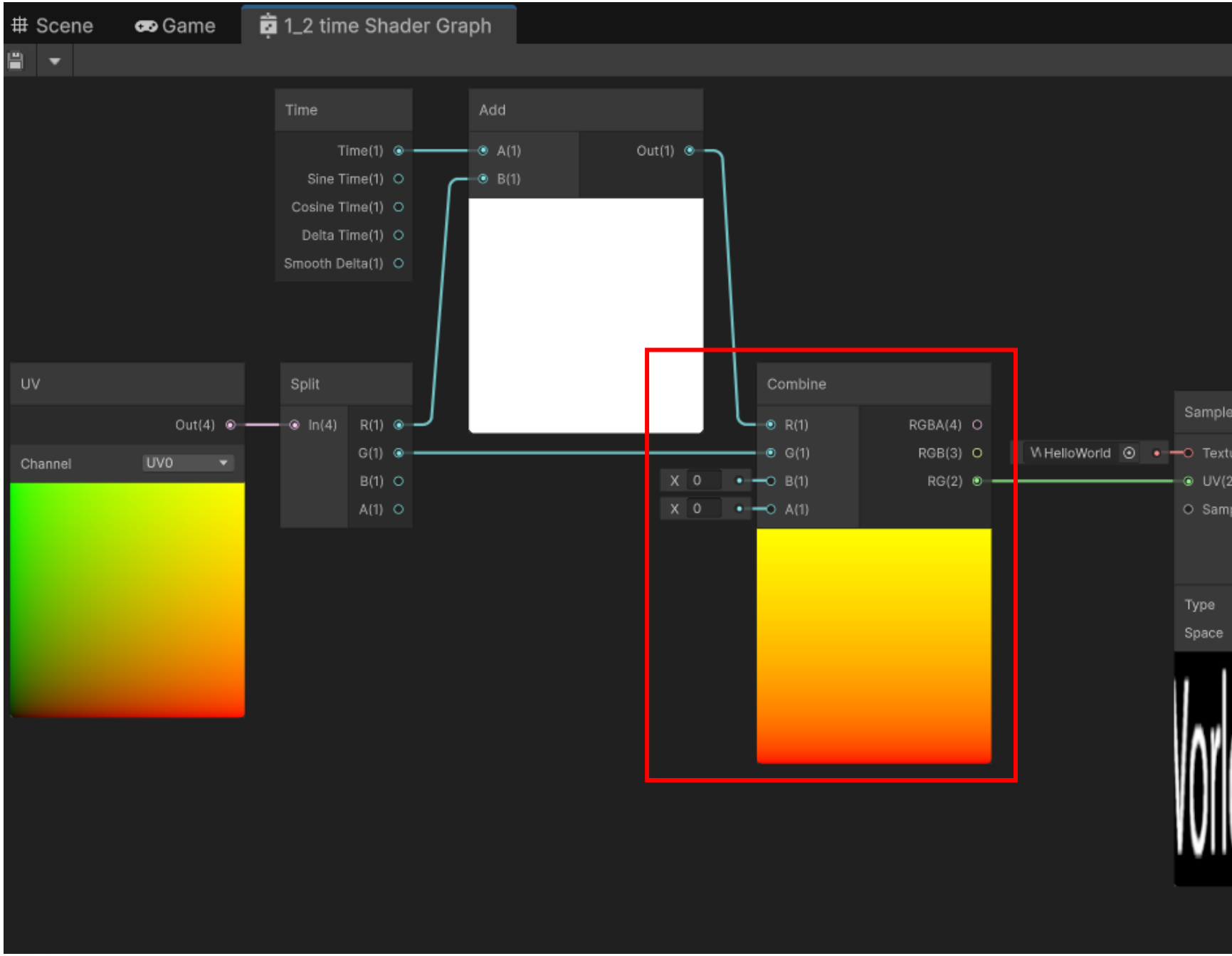
シェーダで回す

- テクスチャ座標をずらす
  - U成分に時間を足して動かす



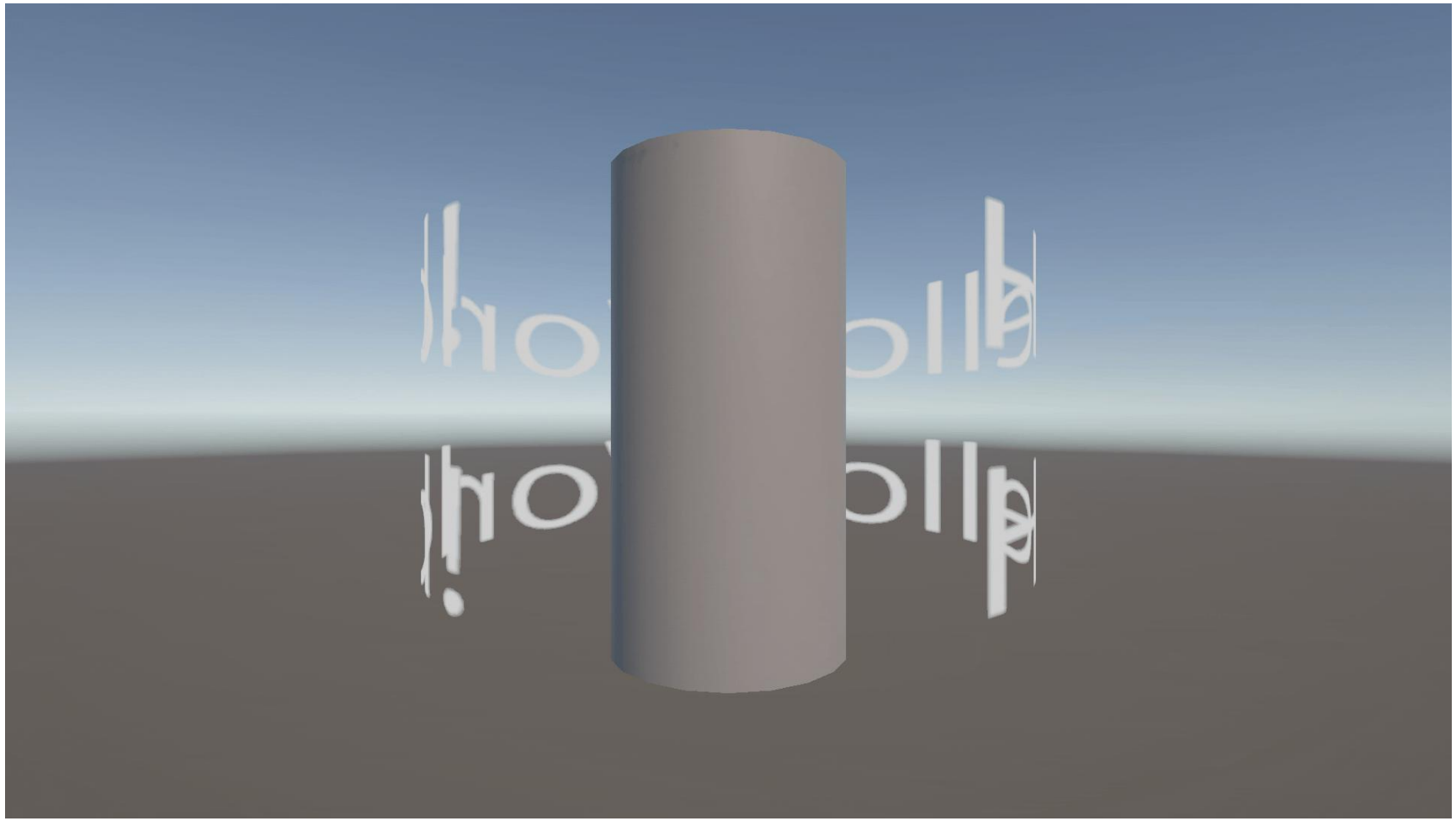
シェーダで回す

- テクスチャ座標を再構築
  - テクスチャ座標は2次元でなければならないので、動かさないV成分と合わせる



やってみよう

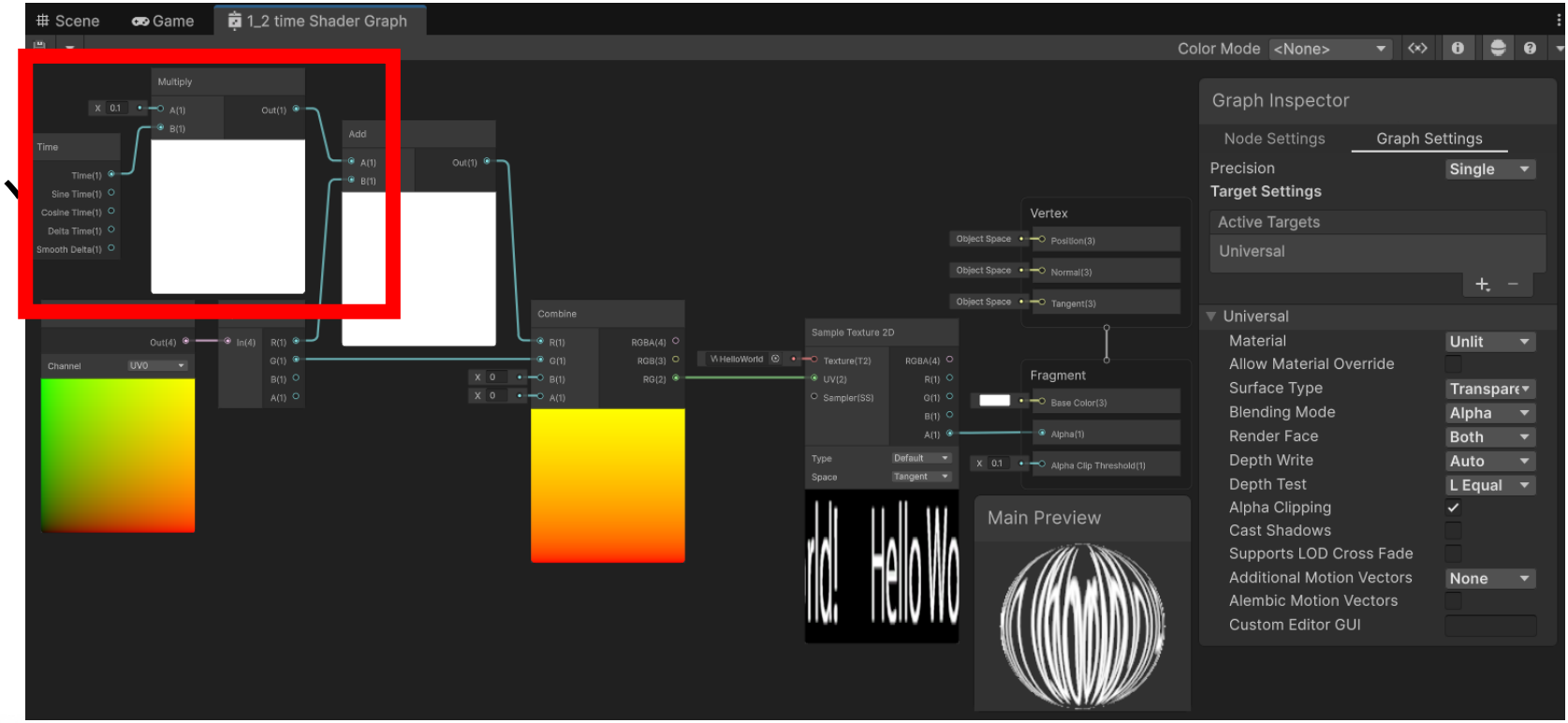
- ただしすごく速い



速度を合わせる

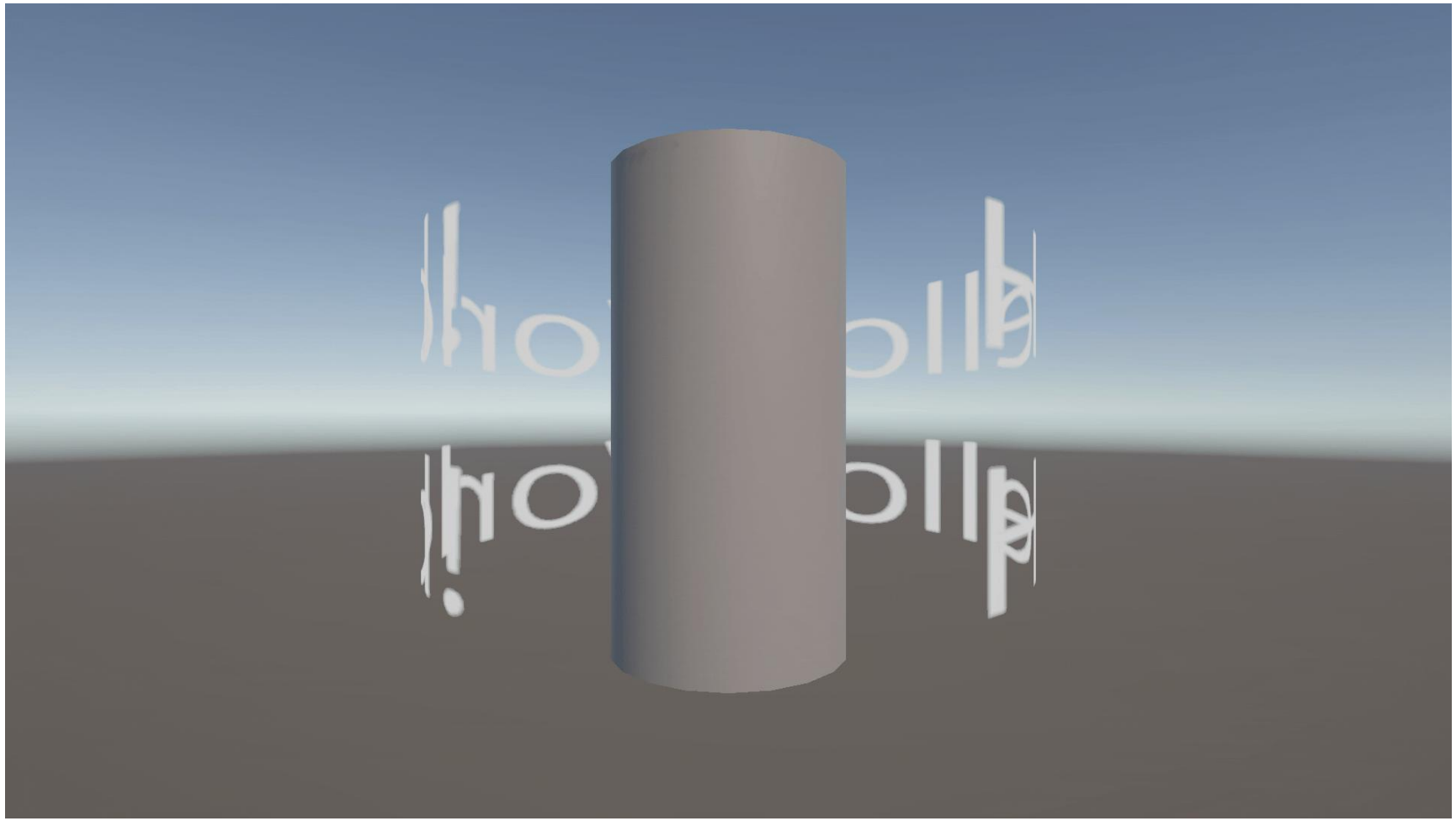
- Timeノードは1秒で1増える
- UVは1.0増えると一周

10秒で一周するには、
10で割る
  - 0.1を掛ける



やってみよう

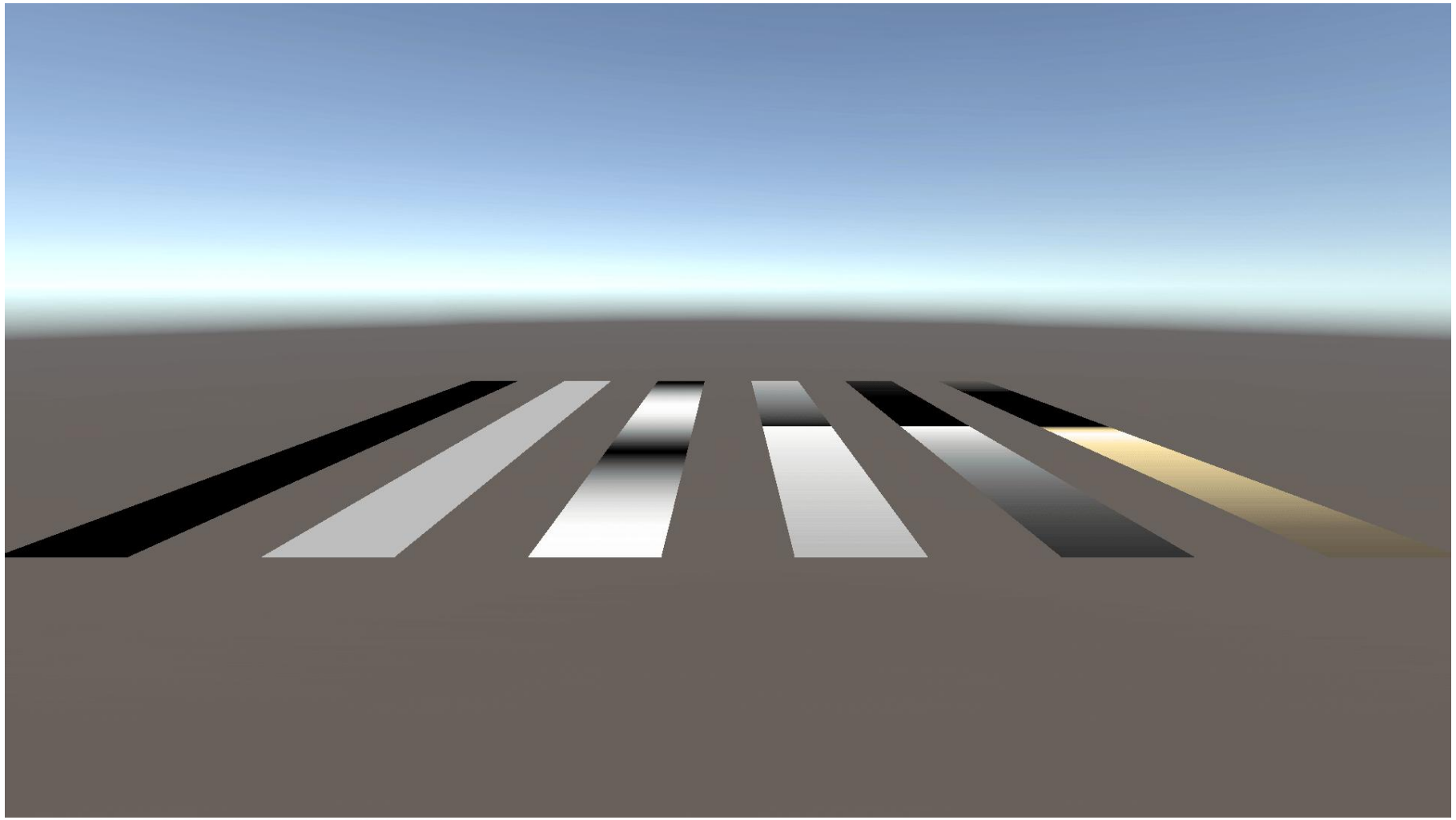
- スクリプトとシェーダで同じ回転が実現できました
- 注意：ゲーム中のdeltaTimeで動くので、ポーズ処理に注意
  - ポーズメニューがあるゲームではCPUから時間を渡して止める



本日の内容

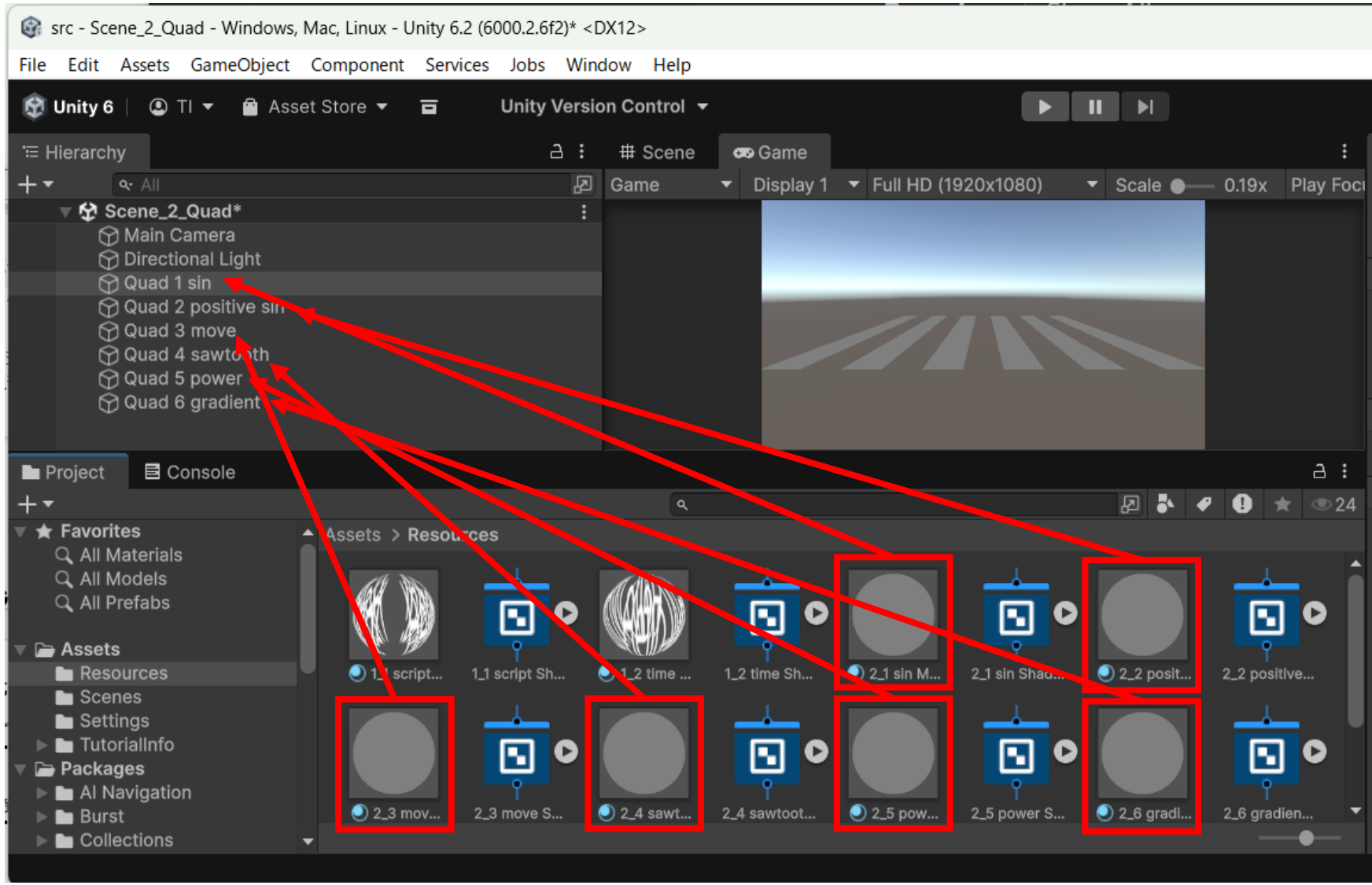
- 光を走らせる
  - 概要
  - 回してみよう
  - 光を走らせてみよう
  - 思った場所に走らせよう

光を走らせてみよう



シーン

- Scene_2_Quad
- マテリアルを付け忘れていたので、設定してください



光を走らせたい

- ずっと走らせる
  - ある程度すると、同じ状態に戻って繰り返し

- 簡単な繰り返しは？
  - 三角関数！

本日の内容

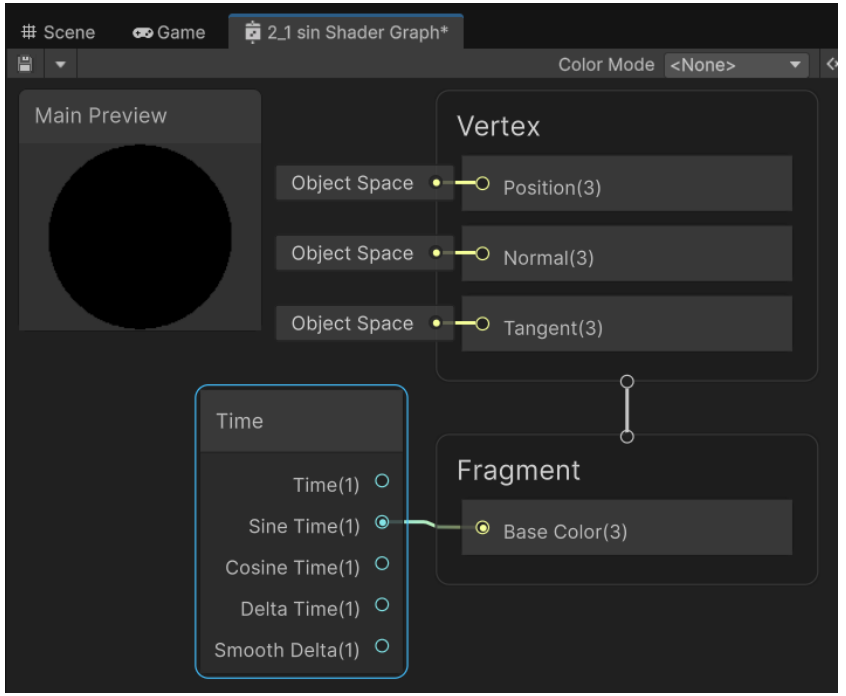
- 光を走らせる
  - 概要
  - 回してみよう
  - 光を走らせてみよう
    - Sin波
  - 思った場所に走らせよう

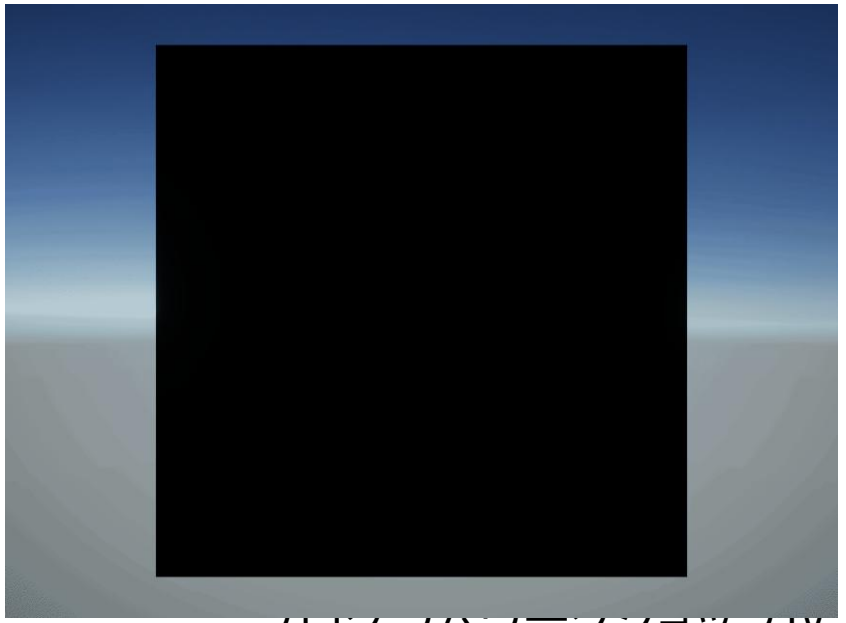
Sine Time
（Cosine Timeもほぼ同様）

- 三角関数っぽいので付けてみる

  sin（経過秒数）

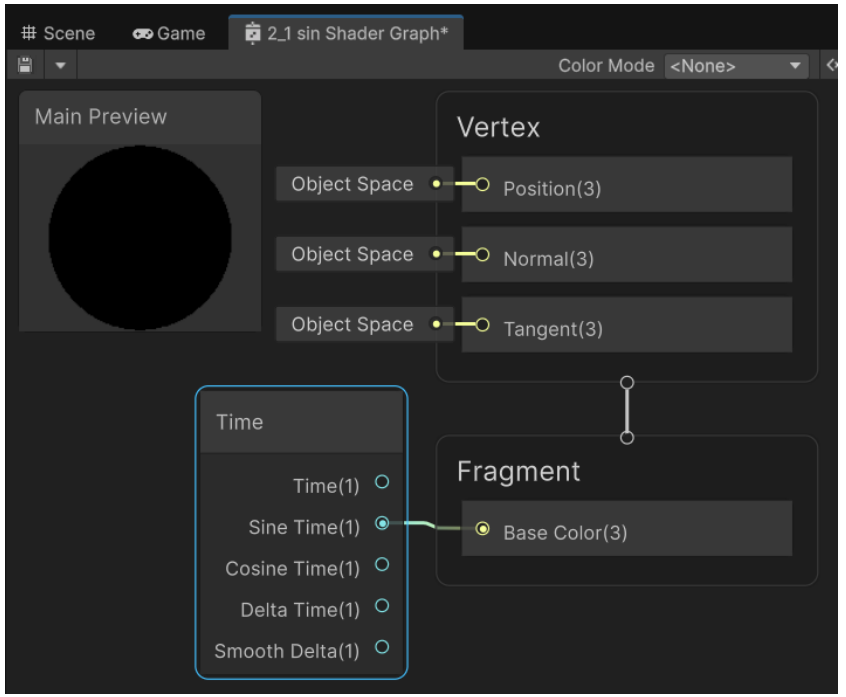
- 2π（≈6.28）秒で元に戻る
  - 周期は変えられない

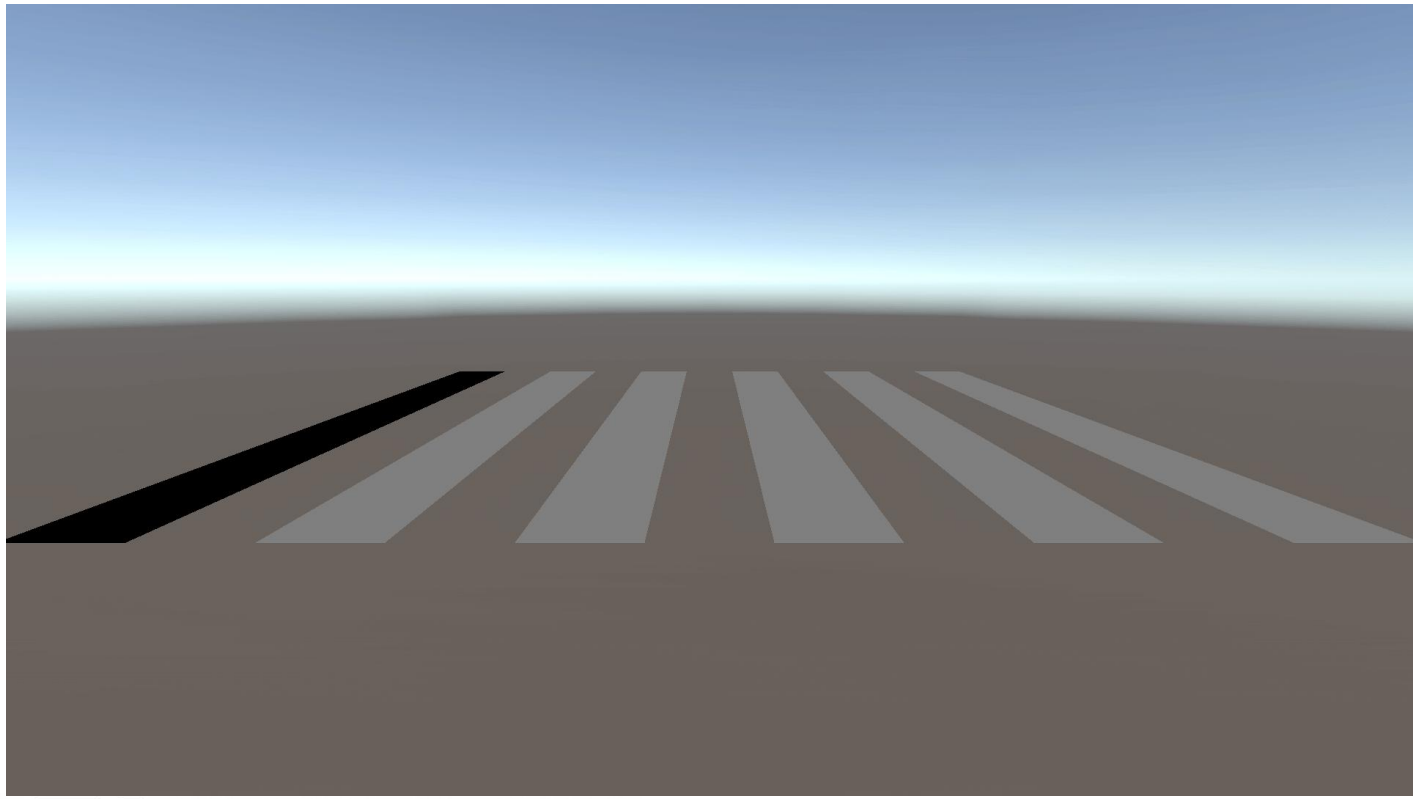




やってみよう

- 「2_1 sin Shader Graph」を
  この状態にしてください



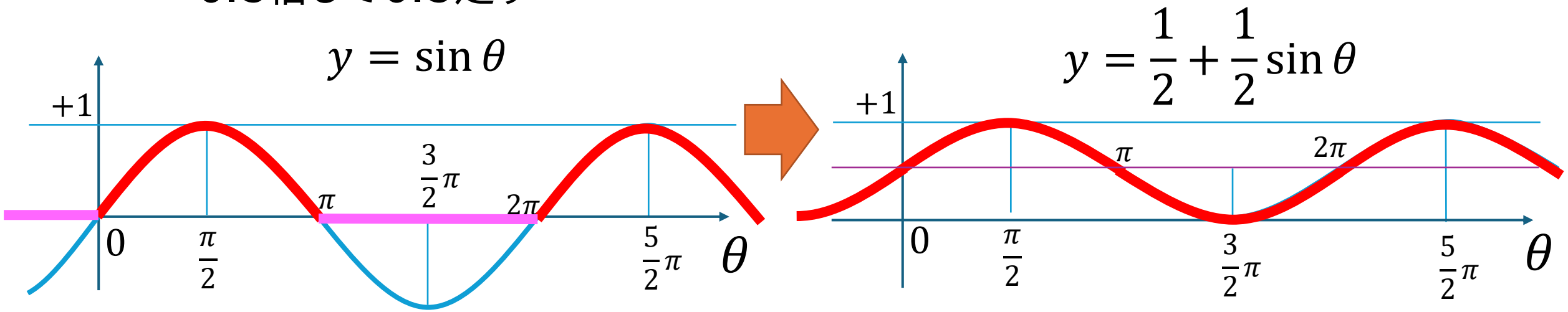


本日の内容

- 光を走らせる
  - 概要
  - 回してみよう
  - 光を走らせてみよう
    - Sin波
    - 滑らかなSin波
  - 思った場所に走らせよう

Sin関数を色に使う際の問題点

- 表示されるのは正の値
  - 半分の時間は黒
- 負の値を消すには、[-1,+1]の範囲を[0,+1]に縮めれば良い
  - 0.5倍して0.5足す



Scene Game 2_2 positive sin Shader Graph

Color Mode

Main Preview

Vertex

Object Space Position(3)

Object Space Normal(3)

Object Space Tangent(3)

Fragment

Base Color(3)

Add

X 0.5 A(1) B(1) Out(1)

Multiply

X 0.5 A(1) B(1) Out(1)

Time

Time(1)

Sine Time(1)

Cosine Time(1)

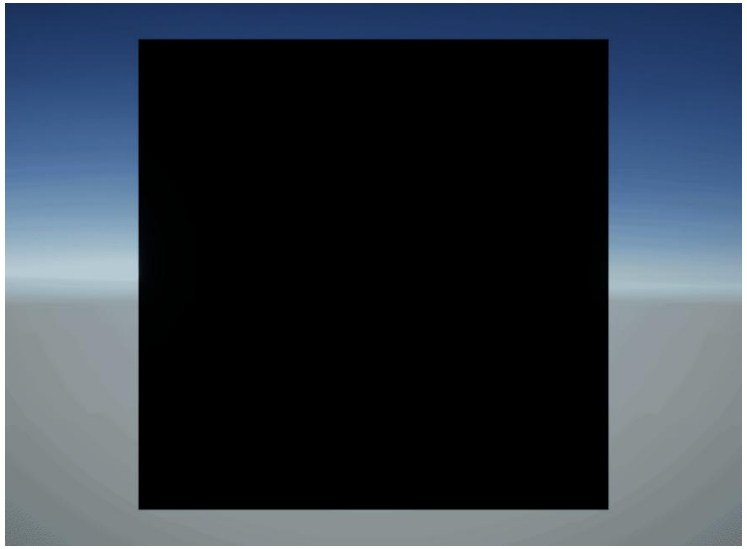
Delta Time(1)

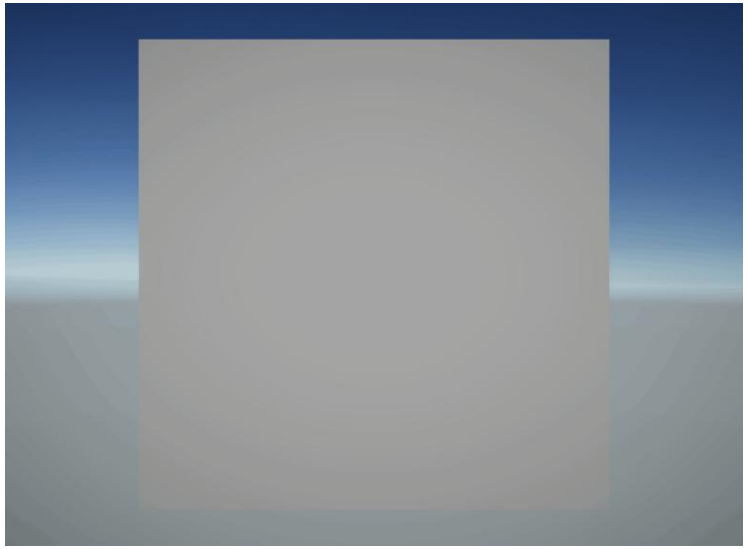
Smooth Delta(1)

$\times \frac{1}{2}$

$+ \frac{1}{2}$

$y = \sin\theta$

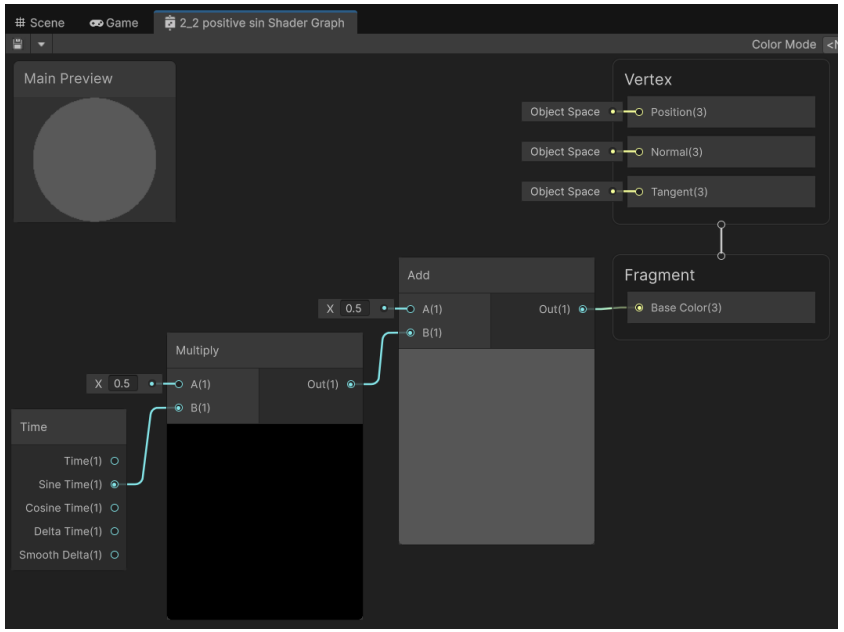


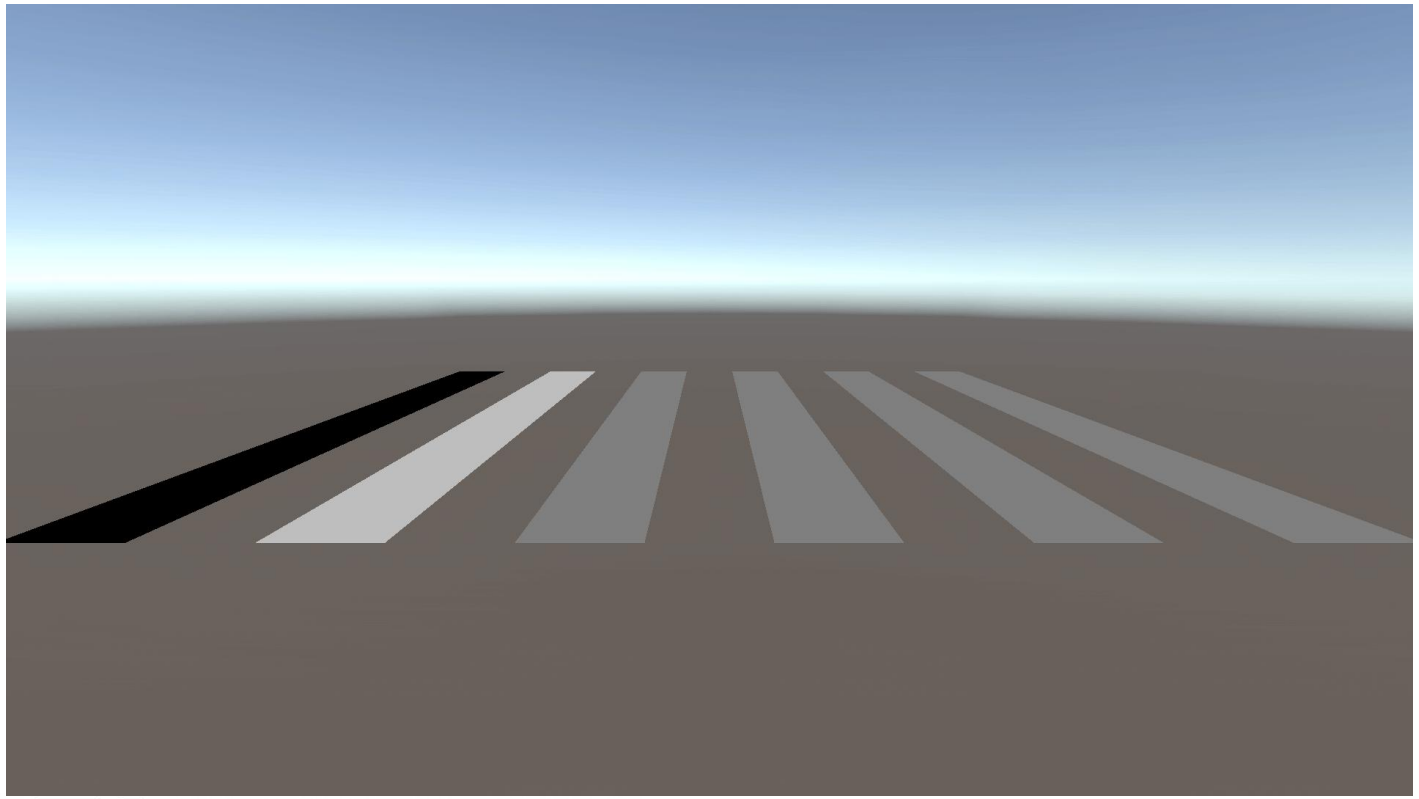


$y = \frac{1}{2} + \frac{1}{2}\sin\theta$

やってみよう

- 「2_2 positive sin Shader Graph」をこの状態にしてください



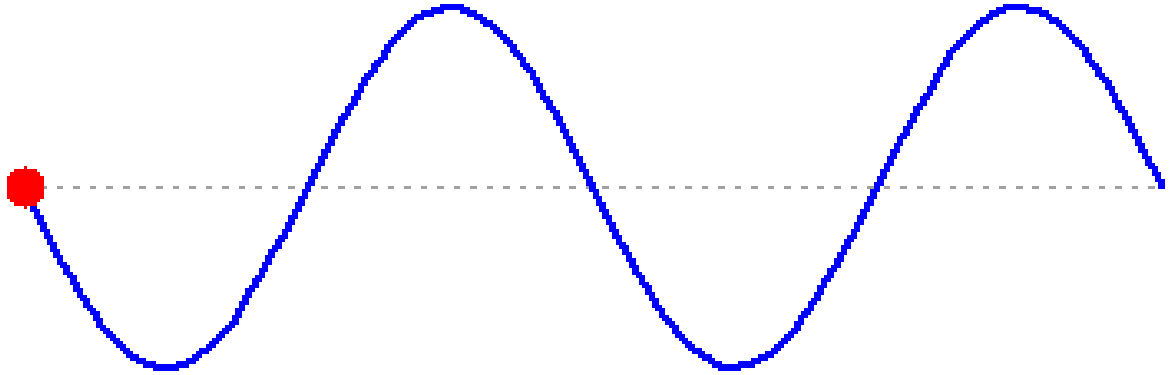


本日の内容

- 光を走らせる
  - 概要
  - 回してみよう
  - 光を走らせてみよう
    - Sin波
    - 滑らかなSin波
    - 進行波
  - 思った場所に走らせよう

次の挑戦：空間的に進ませたい

- 場所ごとに高さが変わる
  - 波動現象



- 波動は波動方程式で記述される

$$\frac{1}{c^2}\frac{\partial^2 u}{\partial t^2} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}$$

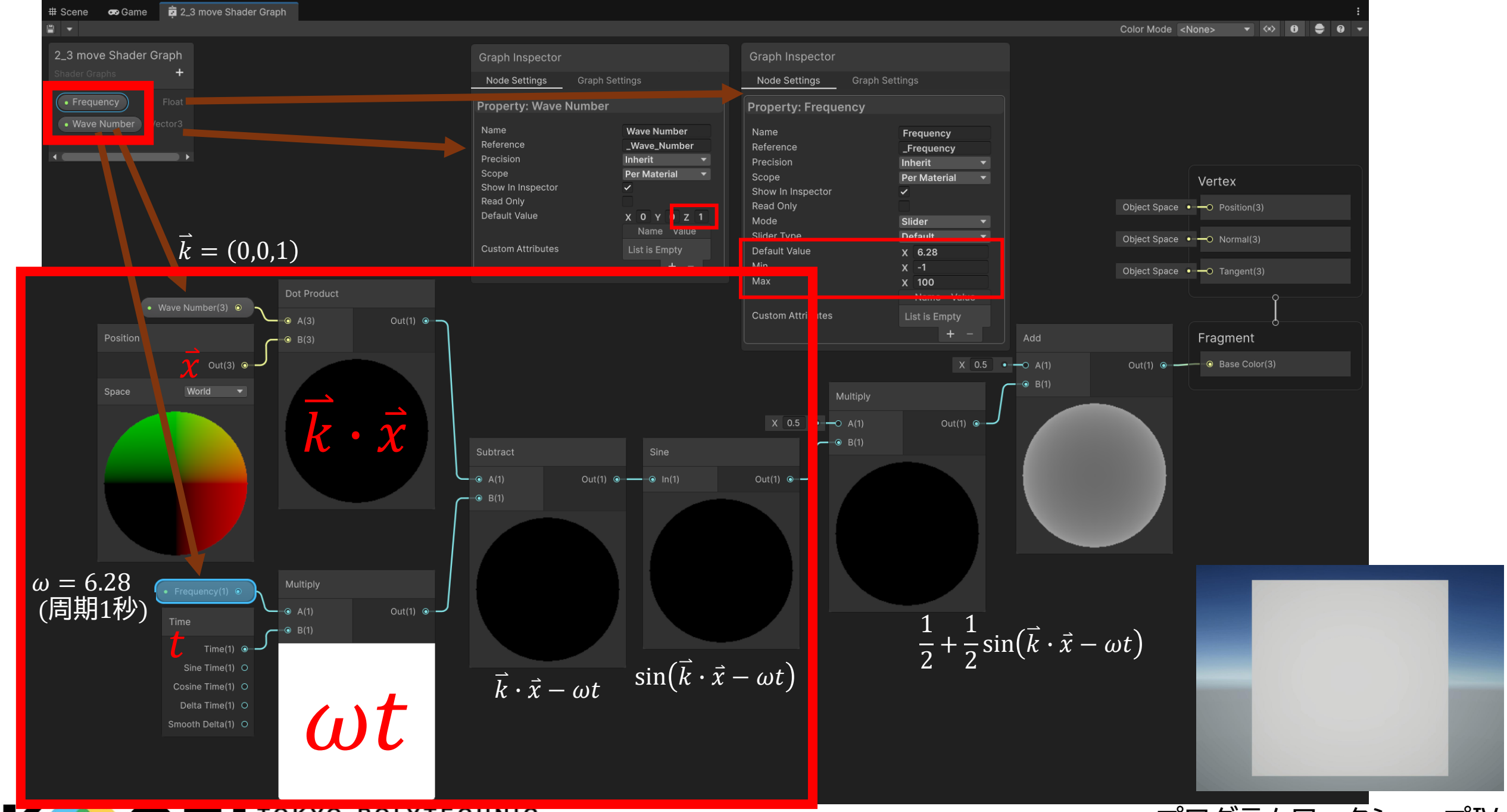
  - 波動方程式の解（ダランベールの解）

$$u = u_f\left(\vec{k}\cdot\vec{x} - \omega t\right) + u_b\left(\vec{k}\cdot\vec{x} + \omega t\right)$$

前進波　　後退波

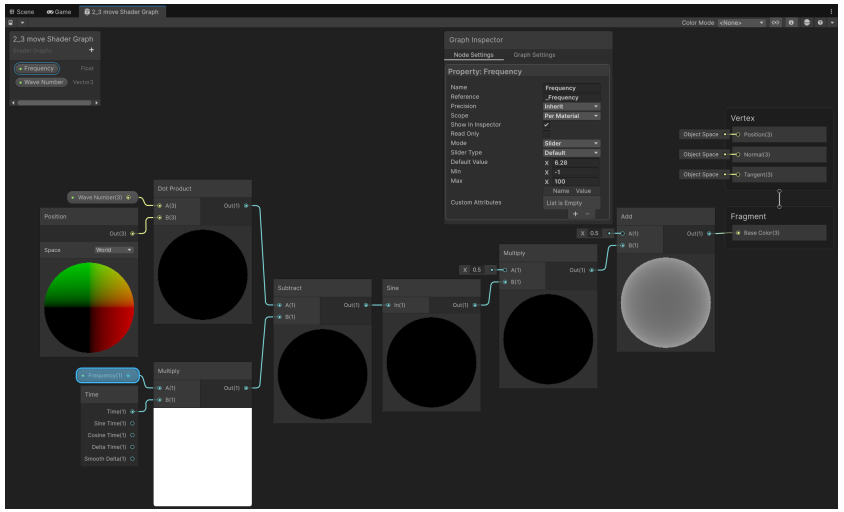
$\sin(t)$　　から

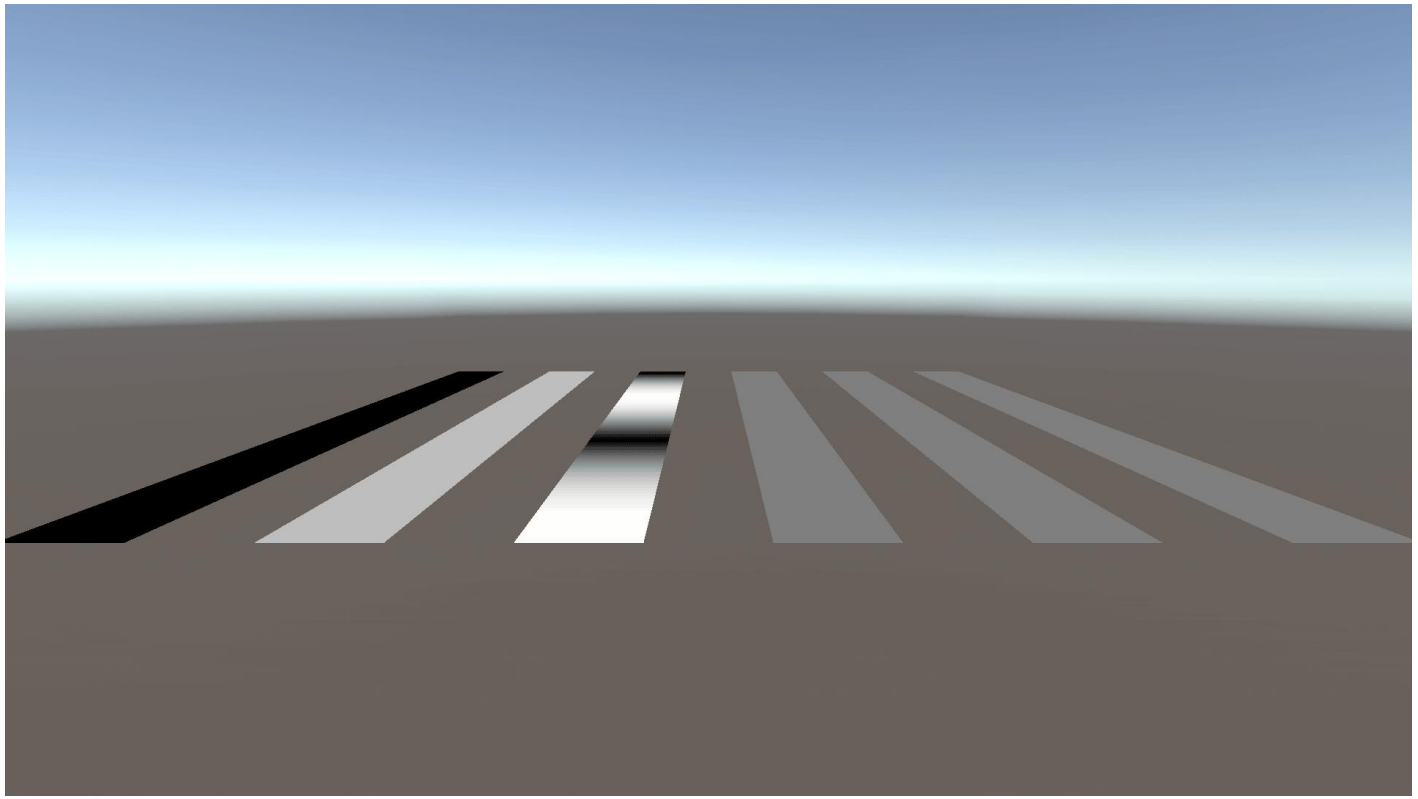
$\sin\left(\vec{k}\cdot\vec{x} - \omega t\right)$　　に変えれば良い


Scene
Game
2_3 move Shader Graph
Color Mode <None>
2_3 move Shader Graph
Shader Graphs
Frequency
Float
Wave Number
Vector3
Graph Inspector
Node Settings
Graph Settings
Property: Wave Number
Name
Wave Number
Reference
_Wave_Number
Precision
Inherit
Scope
Per Material
Show In Inspector
Read Only
Default Value
X 0 Y 0 Z 1
Name Value
Custom Attributes
List is Empty
Graph Inspector
Node Settings
Graph Settings
Property: Frequency
Name
Frequency
Reference
_Frequency
Precision
Inherit
Scope
Per Material
Show In Inspector
Read Only
Mode
Slider
Default Value
X 6.28
X -1
Max
X 100
Custom Attributes
List is Empty
Vertex
Object Space
Position(3)
Object Space
Normal(3)
Object Space
Tangent(3)
Fragment
Base Color(3)
$\vec{k} = (0,0,1)$
Wave Number(3)
Dot Product
A(3)
B(3)
Out(1)
Position
$\vec{x}$
Out(3)
Space
World
$\vec{k} \cdot \vec{x}$
X 0.5
Add
A(1)
B(1)
Out(1)
Multiply
Subtract
Sine
In(1)
$\omega = 6.28$
(周期1秒)
Frequency(1)
Time
t
Time(1)
Sine Time(1)
Cosine Time(1)
Delta Time(1)
Smooth Delta(1)
ωt
$\vec{k} \cdot \vec{x} - \omega t$
$\sin(\vec{k} \cdot \vec{x} - \omega t)$
$\frac{1}{2} + \frac{1}{2}\sin(\vec{k} \cdot \vec{x} - \omega t)$

やってみよう

- 「2_3 move Shader Graph」を
  この状態にしてください





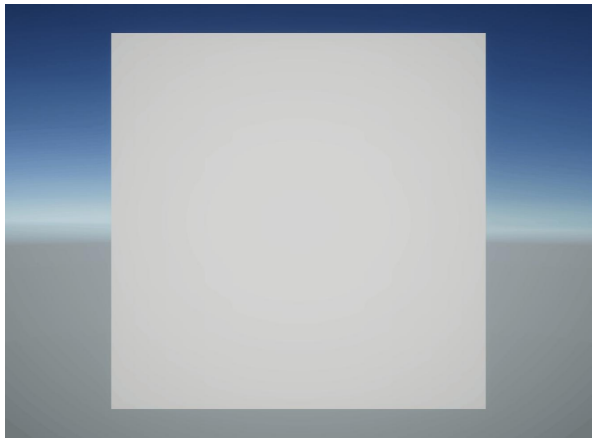
パラメータ： $\frac{1}{2}+\frac{1}{2}\cos\left(\vec{k}\cdot\vec{x}-\omega t\right)$

1. $\vec{k}=(0,0,1),\omega=6.28\ (\sim 2\pi)$

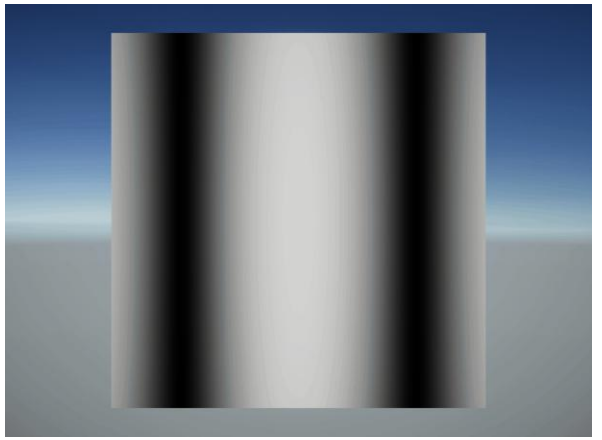
2. $\vec{k}=(0,0,10),\omega=6.28\ (\sim 2\pi)$
   - 波数$\vec{k}$が大きくなると空間的な間隔が狭くなる
     - 波数 $\vec{k}$：単位長さ当たりの波の個数を 2π 倍したもの

3. $\vec{k}=(0,0,1),\omega=15.70\ (\sim 5\pi)$
   - 角振動数ωが大きくなると時間的な間隔が短くなる
     - 速く動く

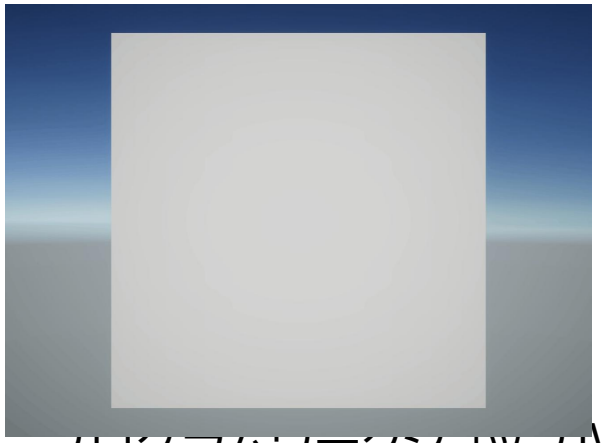
(1)



(2)



(3)



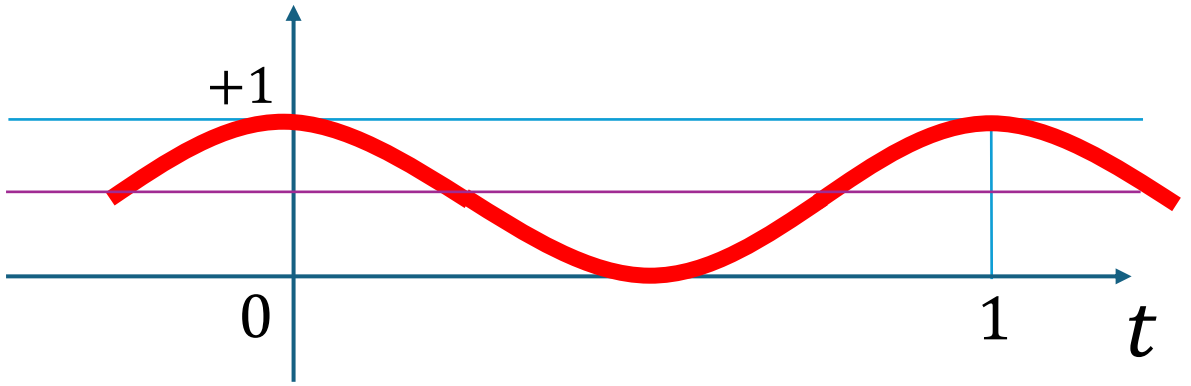
本日の内容

- 光を走らせる
  - 概要
  - 回してみよう
  - 光を走らせてみよう
    - Sin波
    - 滑らかなSin波
    - 進行波
    - のこぎり波
  - 思った場所に走らせよう

挑戦：別の形の波を作りたい

- 余弦関数：なだらか

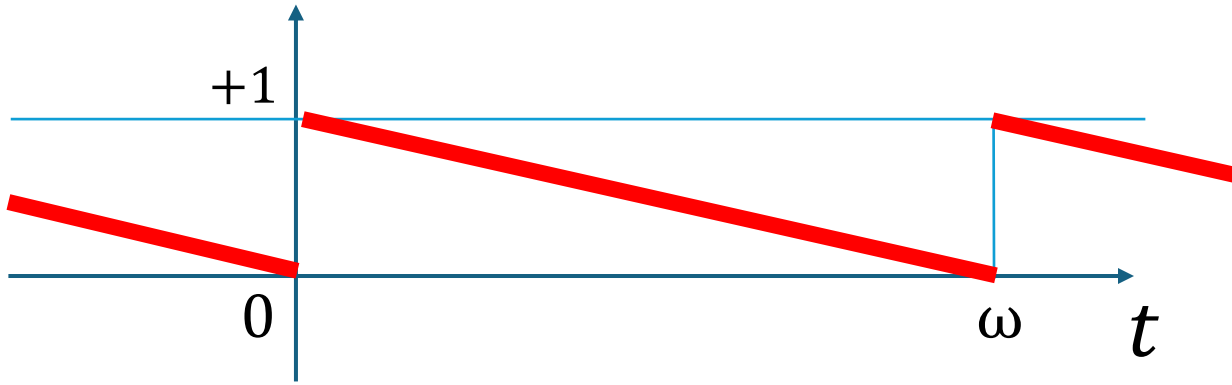
$$y = \frac{1}{2} + \frac{1}{2}\cos 2\pi t$$

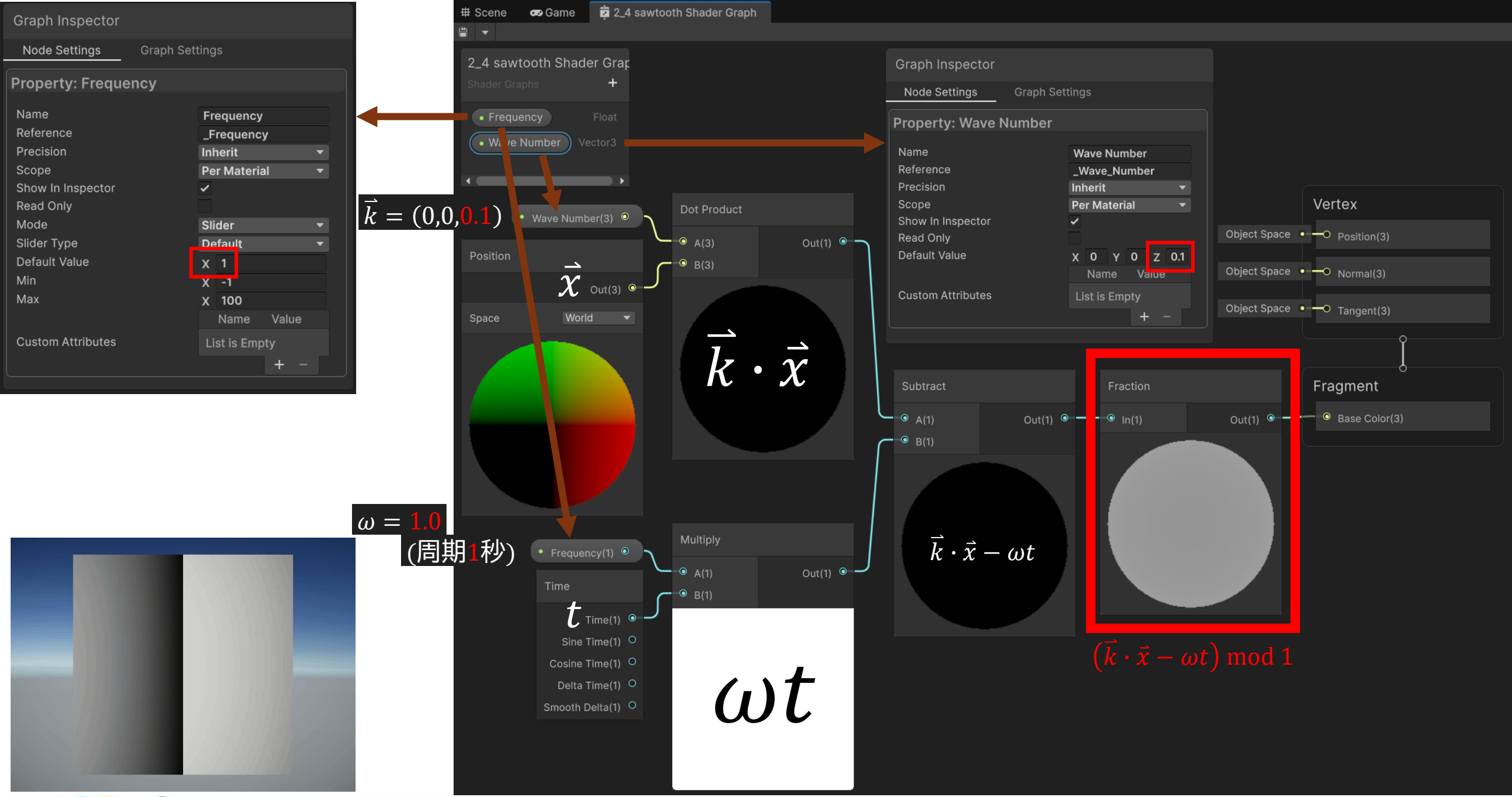


- のこぎり波

$$y = (-\omega t) \bmod 1$$

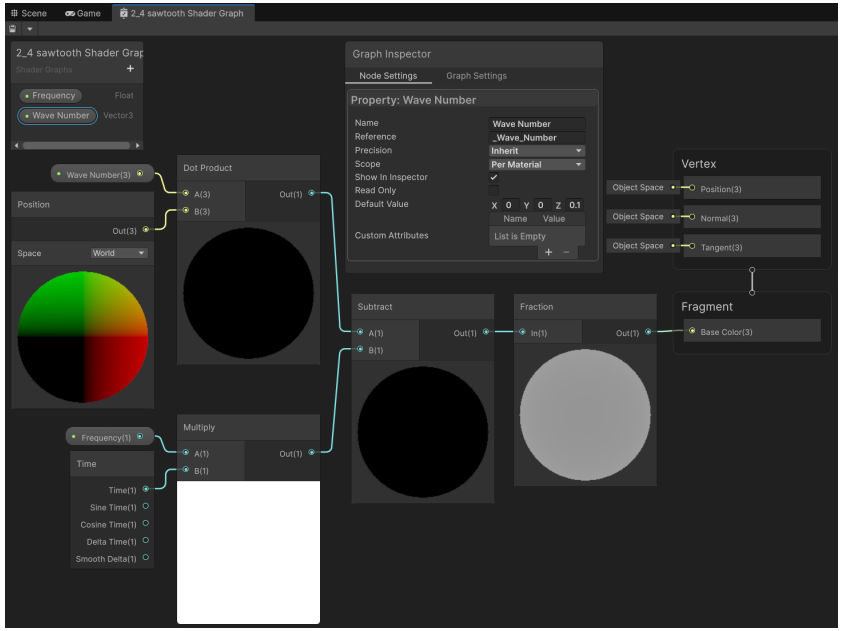
  - $y = (\omega t) \bmod 1$の反対に進む

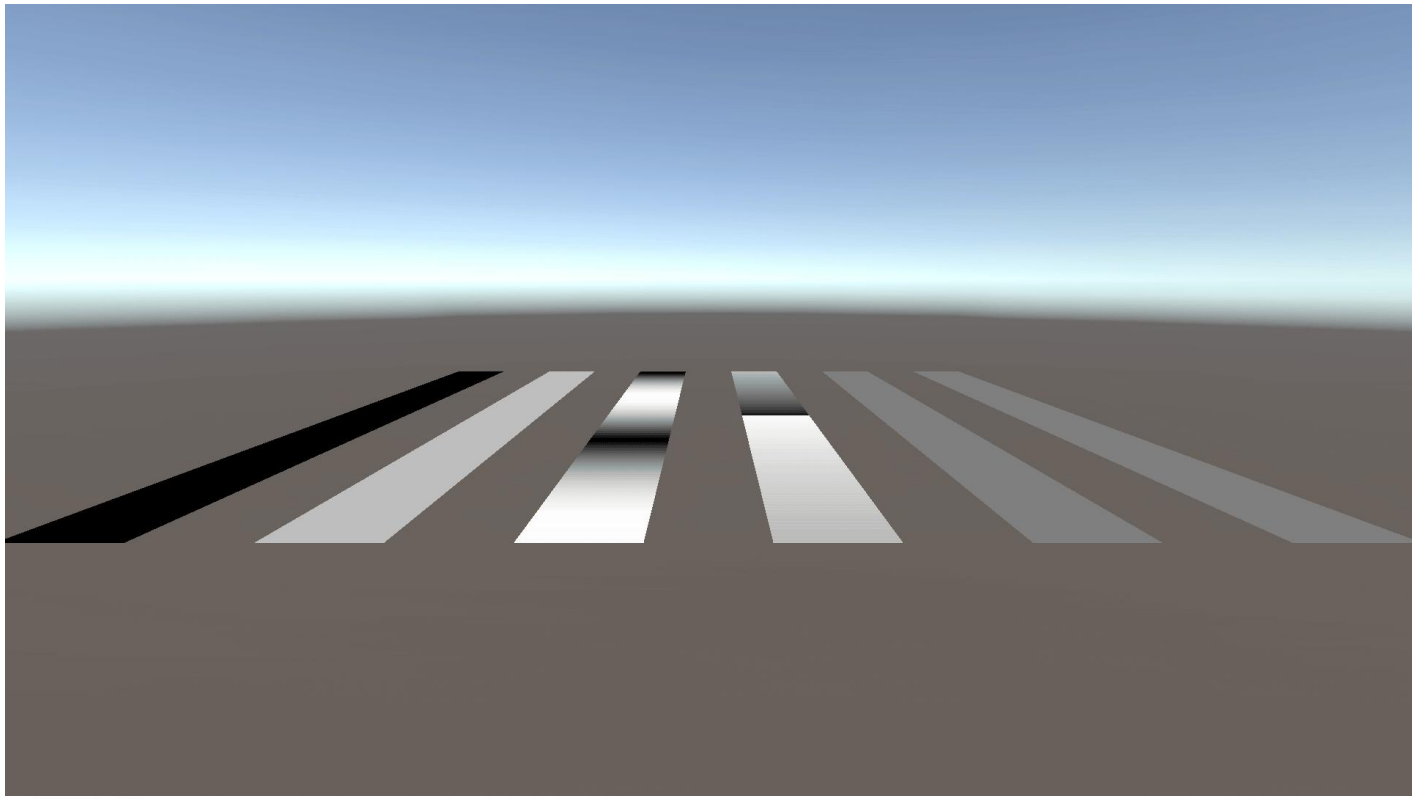



Graph Inspector
Node Settings
Graph Settings
Property: Frequency
Name
Frequency
Reference
_Frequency
Precision
Inherit
Scope
Per Material
Show In Inspector
Read Only
Mode
Slider
Slider Type
Default
Default Value
X 1
Min
X -1
Max
X 100
Name
Value
Custom Attributes
List is Empty
Scene
Game
2_4 sawtooth Shader Graph
2_4 sawtooth Shader Grap
Shader Graphs
Frequency
Float
Wave Number
Vector3
$\vec{k} = (0,0,0.1)$
Wave Number(3)
Position
$\vec{x}$
Out(3)
Space
World
Dot Product
A(3)
B(3)
Out(1)
$\vec{k} \cdot \vec{x}$
Graph Inspector
Node Settings
Graph Settings
Property: Wave Number
Name
Wave Number
Reference
_Wave_Number
Precision
Inherit
Scope
Per Material
Show In Inspector
Read Only
Default Value
X 0 Y 0 Z 0.1
Name
Value
Custom Attributes
List is Empty
Vertex
Object Space
Position(3)
Object Space
Normal(3)
Object Space
Tangent(3)
Fragment
Base Color(3)
Subtract
A(1)
B(1)
Out(1)
$\vec{k} \cdot \vec{x} - \omega t$
Fraction
In(1)
Out(1)
$(\vec{k} \cdot \vec{x} - \omega t) \bmod 1$
$\omega = 1.0$
(周期1秒)
Frequency(1)
Time
t
Time(1)
Sine Time(1)
Cosine Time(1)
Delta Time(1)
Smooth Delta(1)
Multiply
A(1)
B(1)
Out(1)
ωt

やってみよう

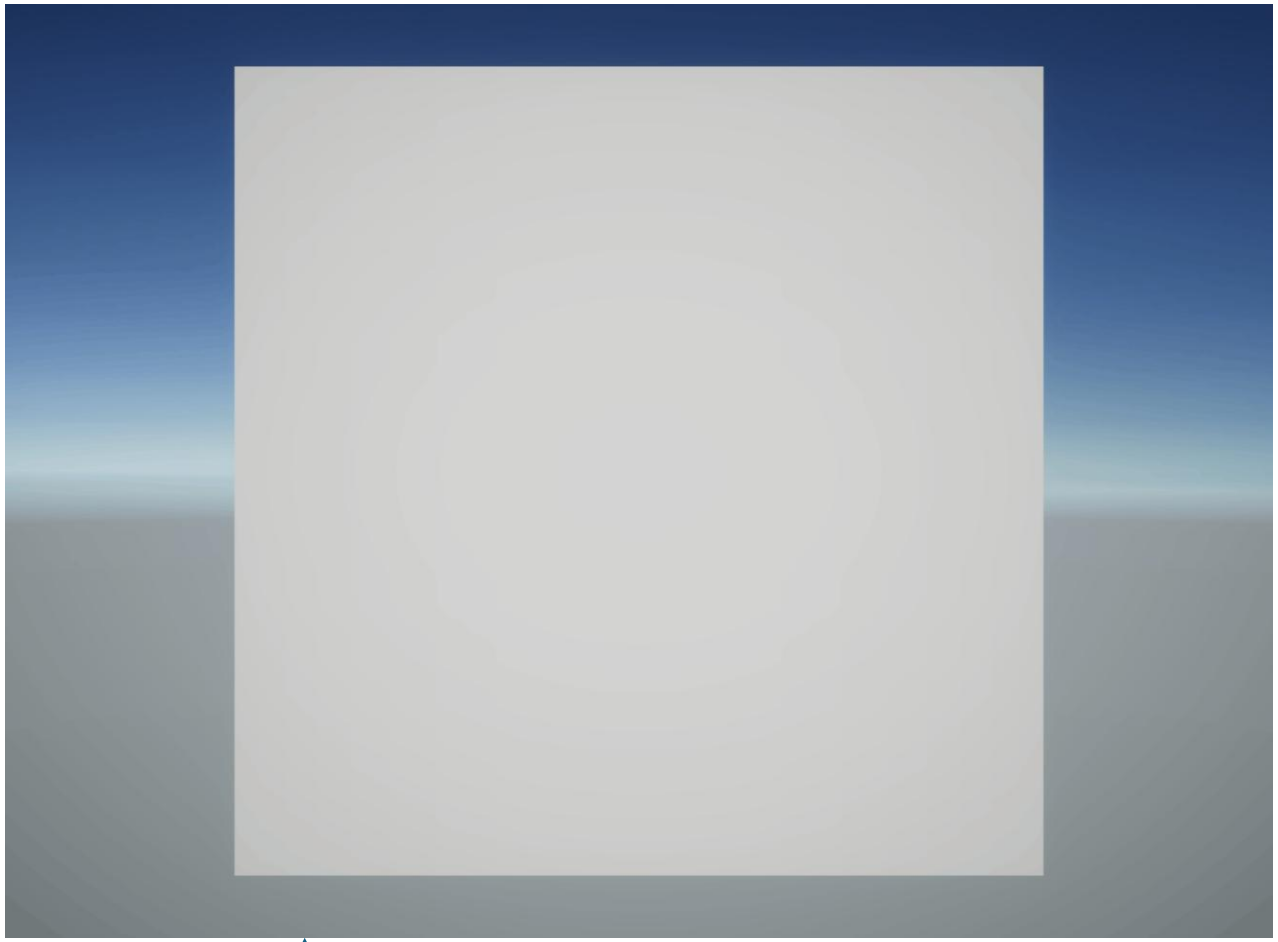
- 「2_4 sawtooth Shader Graph」を
  この状態にしてください

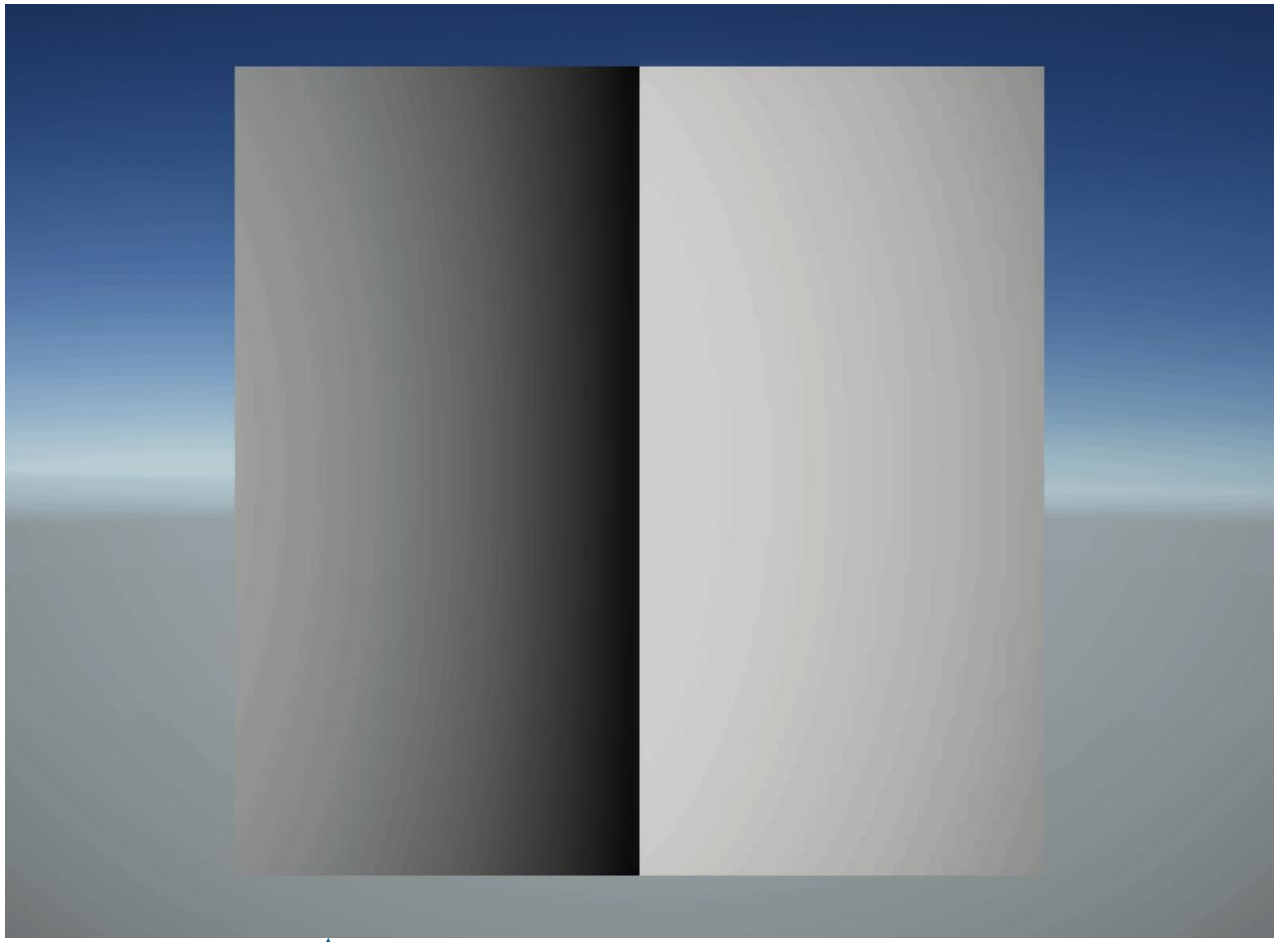


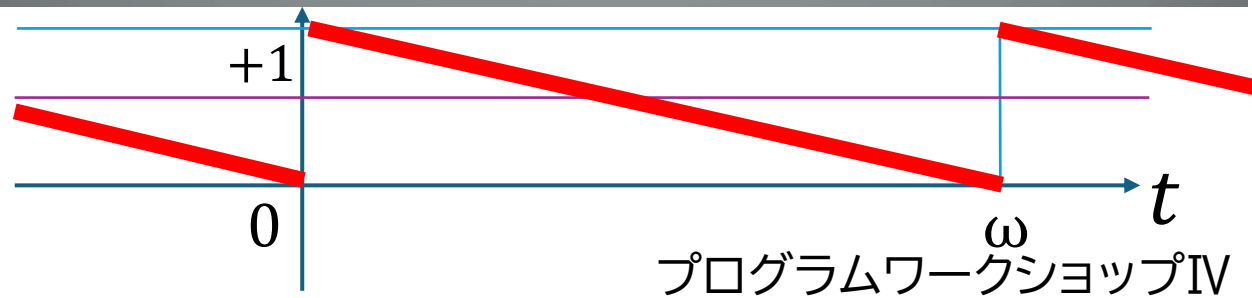


$$y = \frac{1}{2} + \frac{1}{2}\cos 2\pi t$$

$$y = (-\omega t) \bmod 1$$







本日の内容

- 光を走らせる
  - 概要
  - 回してみよう
  - 光を走らせてみよう
    - Sin波
    - 滑らかなSin波
    - 進行波
    - のこぎり波
    - 光のデザイン
  - 思った場所に走らせよう

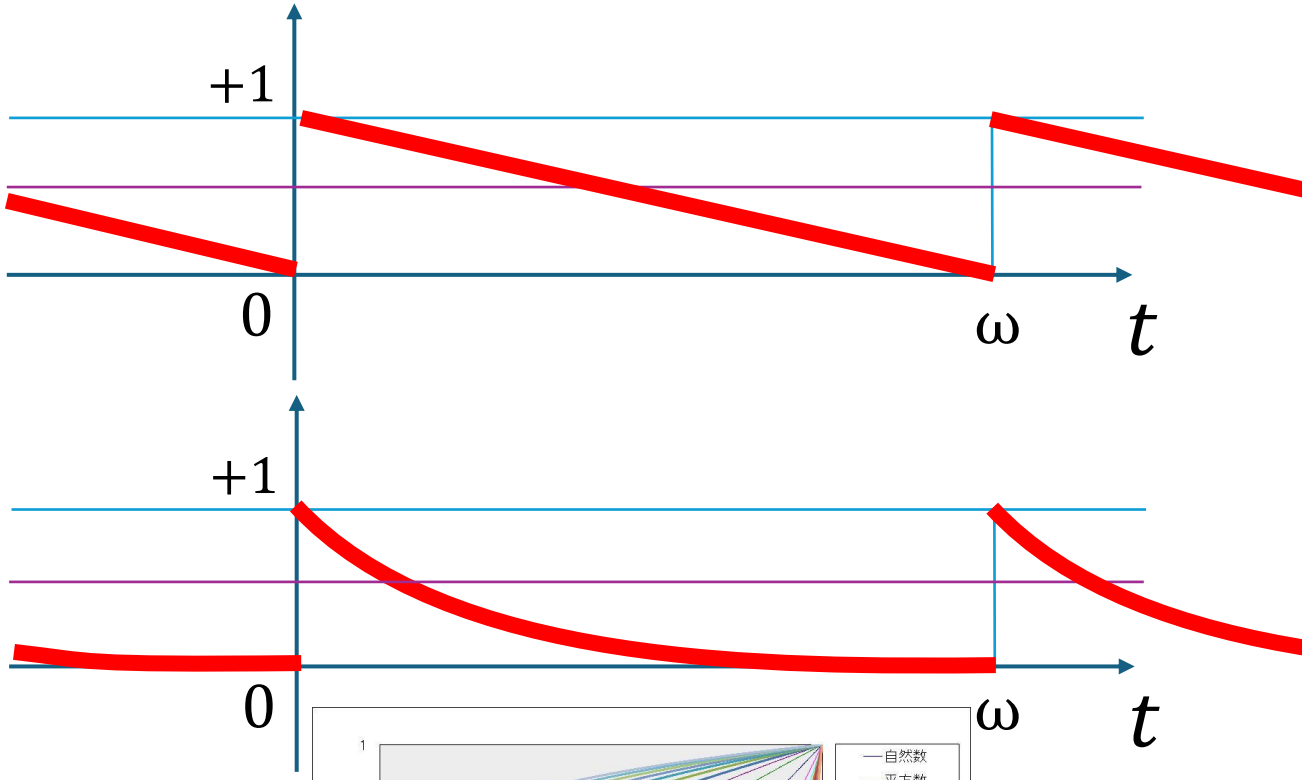
挑戦:もっとデザインしたい

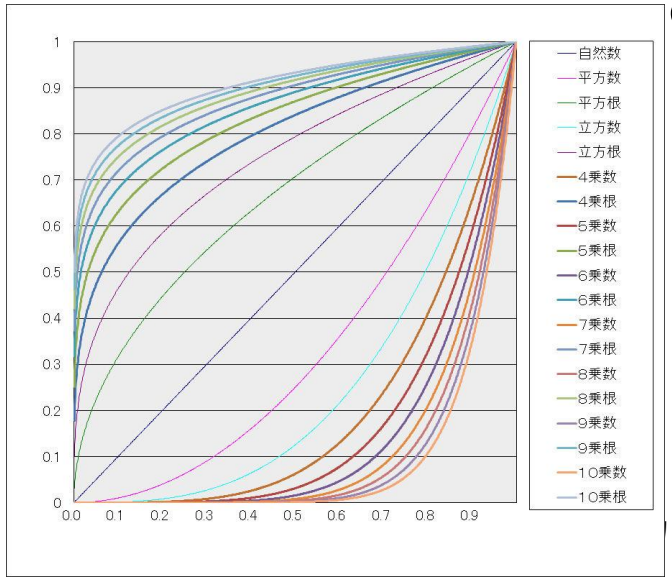
- のこぎり波:突然明るくなる

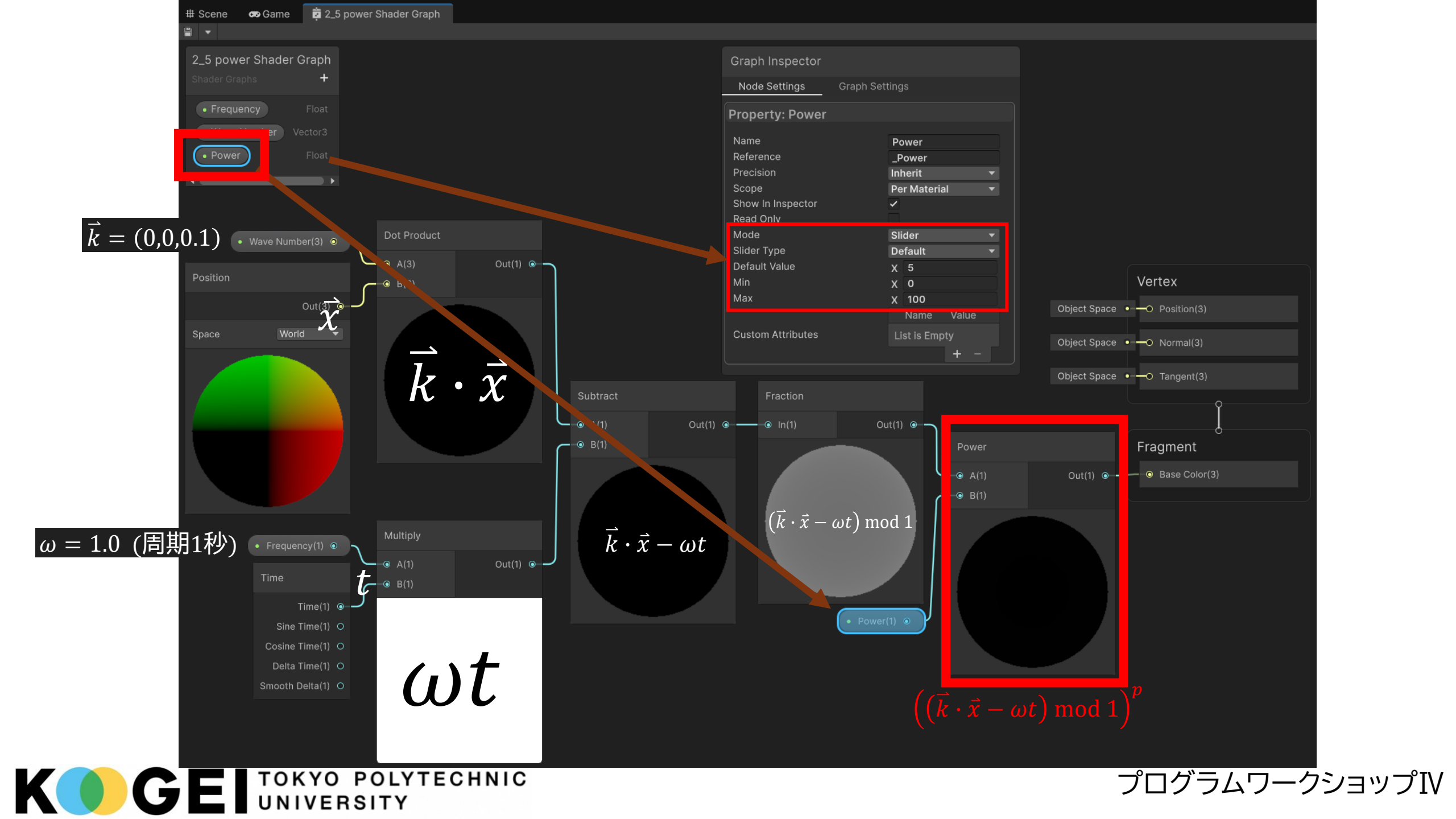
$$y = (-\omega t) \bmod 1$$

- さらにべき乗:すっと消せる

$$y = \left((-\omega t) \bmod 1\right)^{p}$$

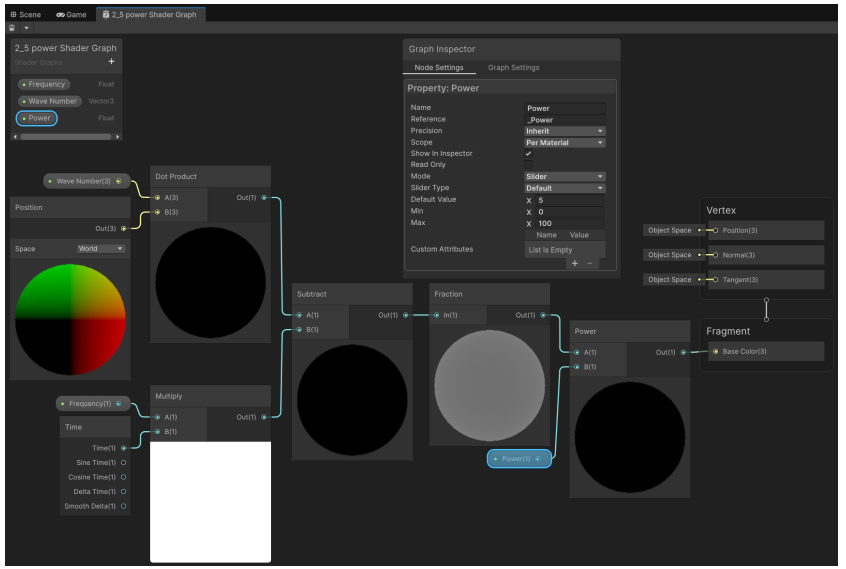


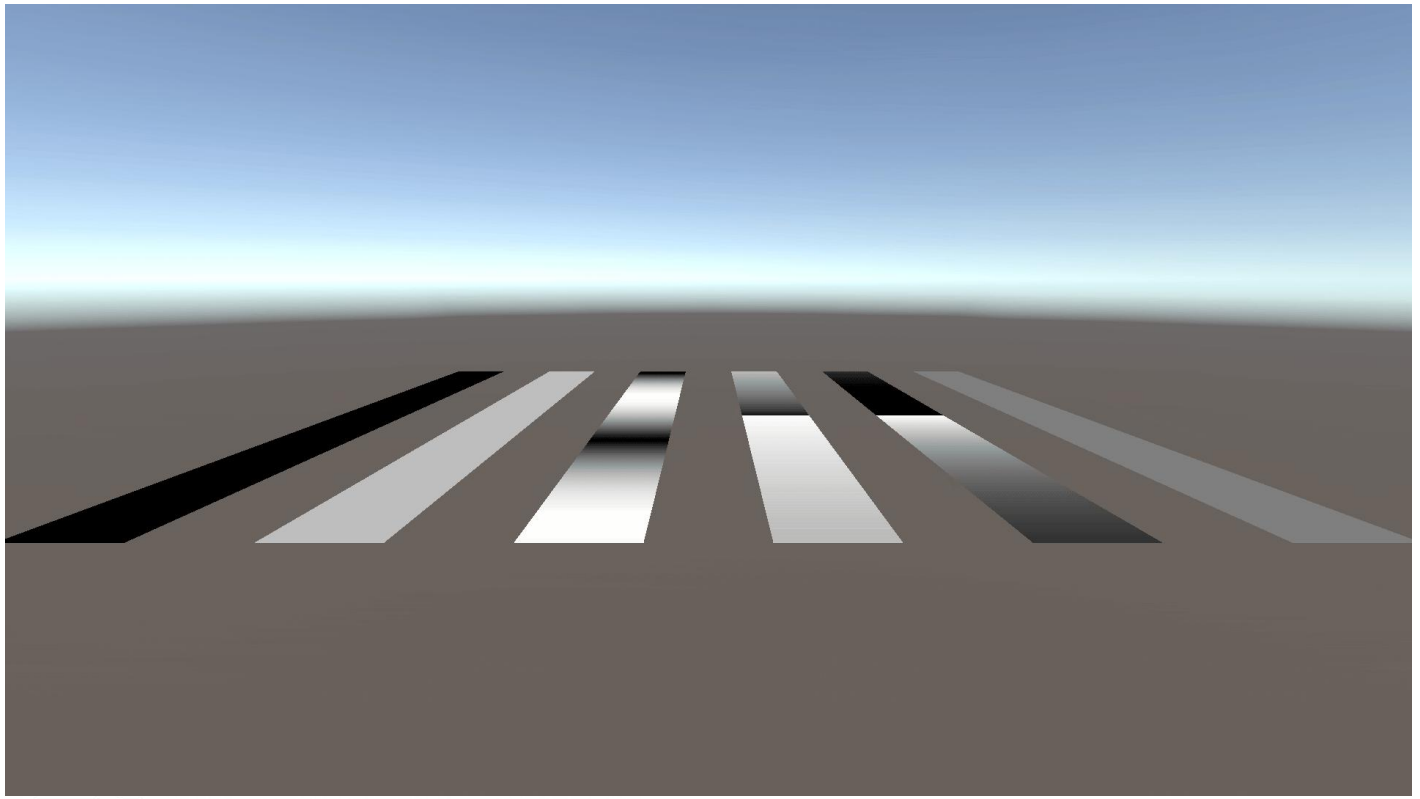



Scene
Game
2_5 power Shader Graph
2_5 power Shader Graph
Shader Graphs
Frequency
Float
Vector3
Power
Float
Graph Inspector
Node Settings
Graph Settings
Property: Power
Name
Power
Reference
_Power
Precision
Inherit
Scope
Per Material
Show In Inspector
Read Only
Mode
Slider
Slider Type
Default
Default Value
X 5
Min
X 0
Max
X 100
Name
Value
Custom Attributes
List is Empty
$\vec{k} = (0,0,0.1)$
Wave Number(3)
Position
Out(3)
$\vec{x}$
Space
World
Dot Product
A(3)
B(3)
Out(1)
$\vec{k} \cdot \vec{x}$
Subtract
A(1)
B(1)
Out(1)
$\vec{k} \cdot \vec{x} - \omega t$
Fraction
In(1)
Out(1)
$(\vec{k} \cdot \vec{x} - \omega t) \bmod 1$
Power(1)
Power
A(1)
B(1)
Out(1)
$\left((\vec{k} \cdot \vec{x} - \omega t) \bmod 1\right)^p$
Vertex
Object Space
Position(3)
Object Space
Normal(3)
Object Space
Tangent(3)
Fragment
Base Color(3)
$\omega = 1.0$ (周期1秒)
Frequency(1)
Time
Time(1)
Sine Time(1)
Cosine Time(1)
Delta Time(1)
Smooth Delta(1)
t
Multiply
A(1)
B(1)
Out(1)
ωt

やってみよう

- 「2_5 power Shader Graph」を
  この状態にしてください

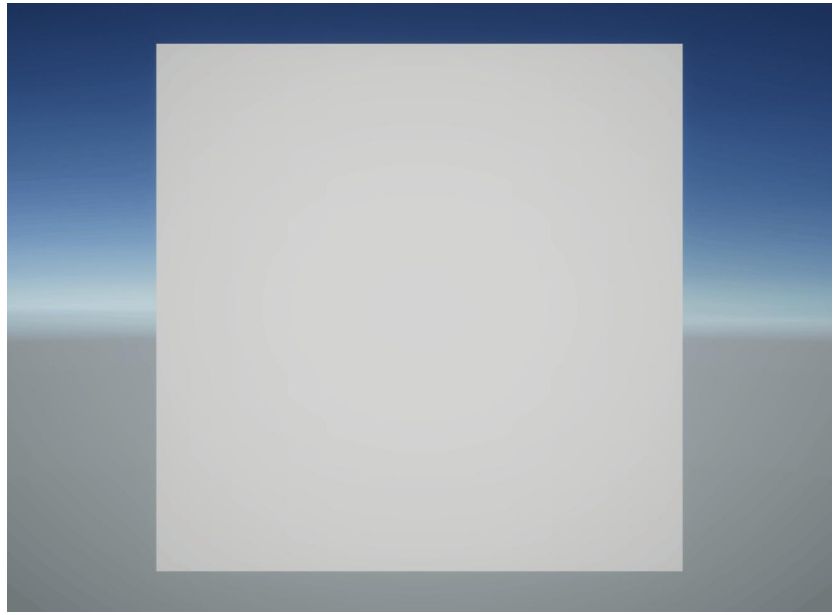


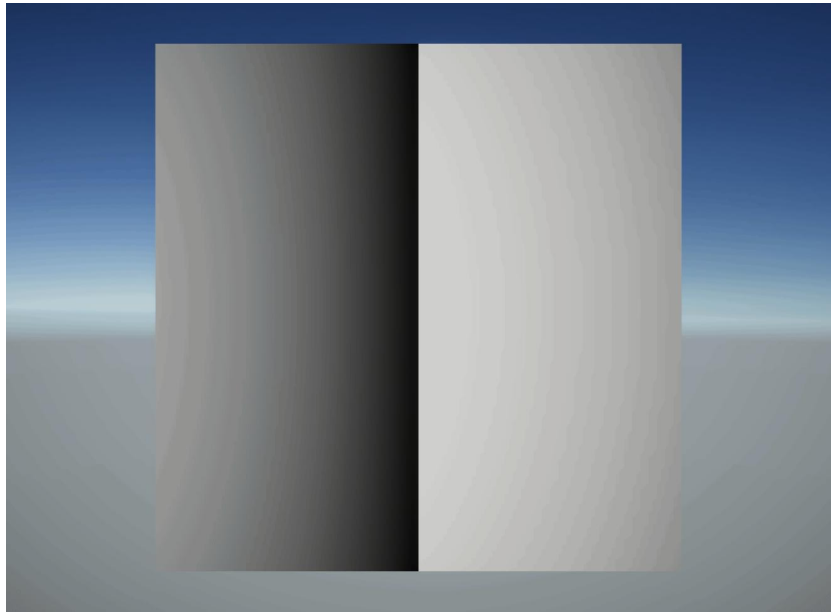


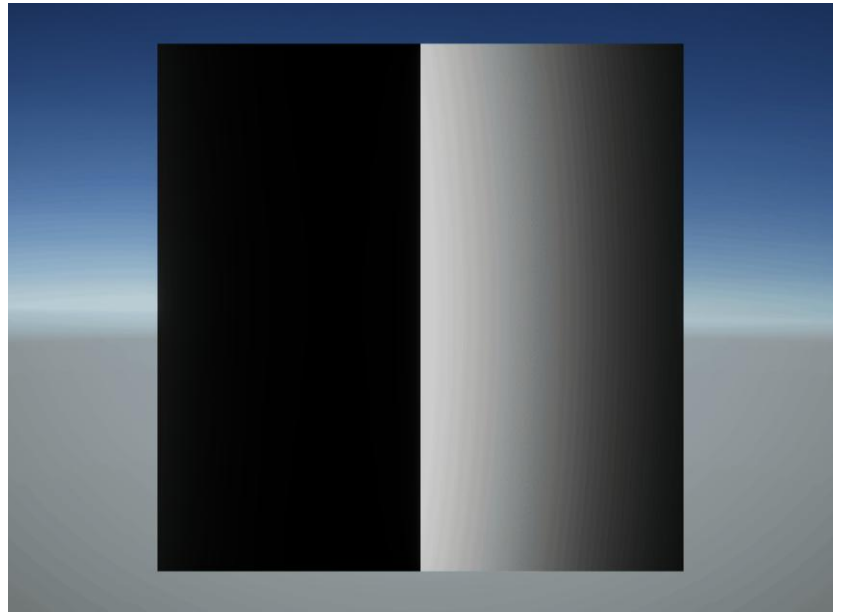
$$y = \frac{1}{2} + \frac{1}{2}\cos 2\pi t$$

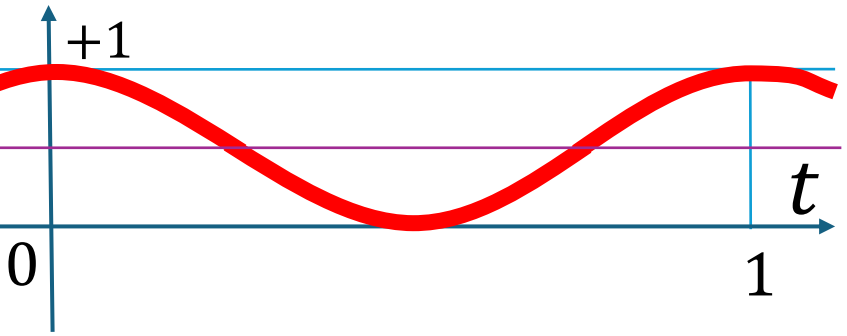
$$y = (-\omega t) \bmod 1$$

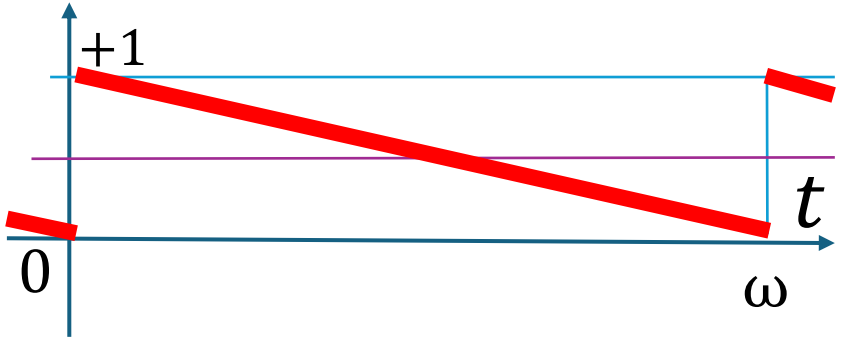
$$y = \left((-\omega t) \bmod 1\right)^5$$

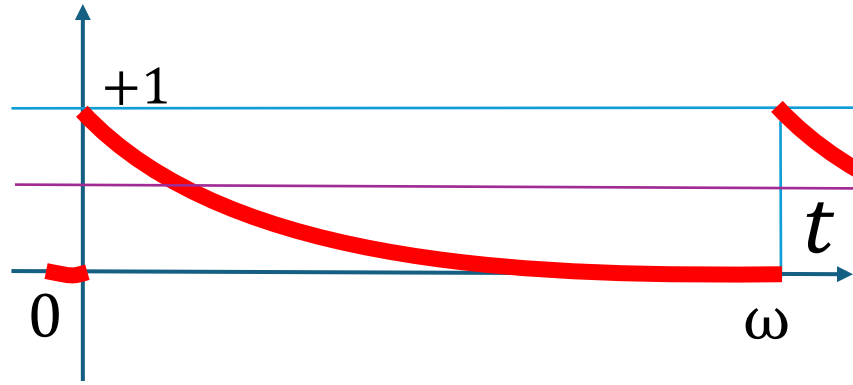










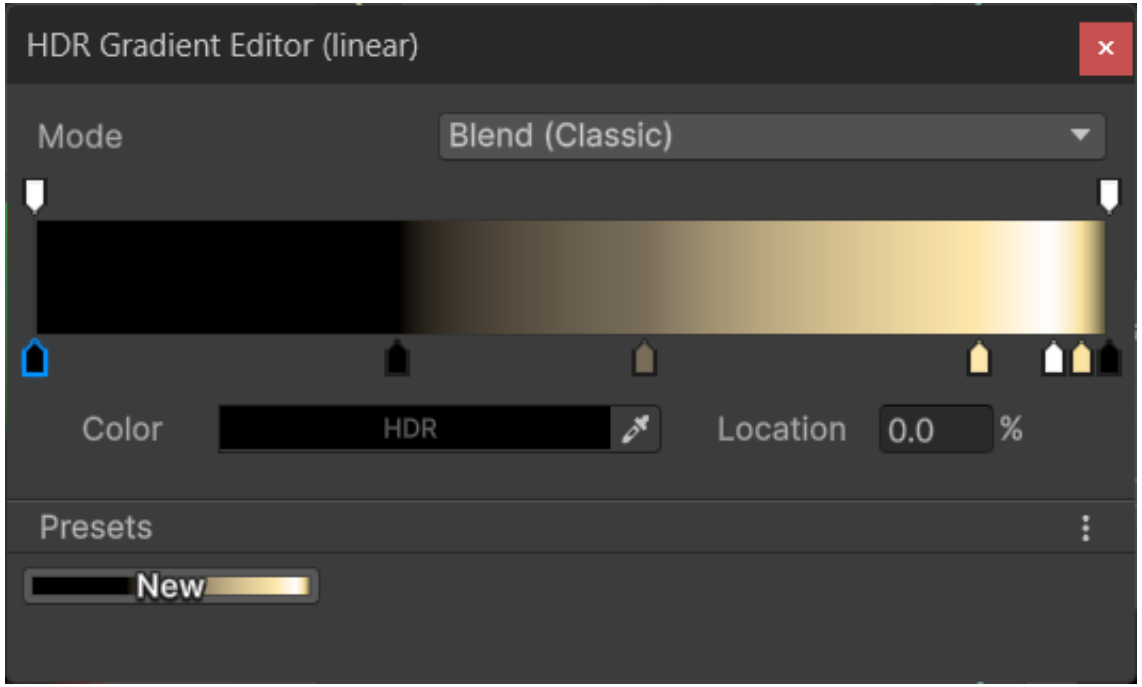


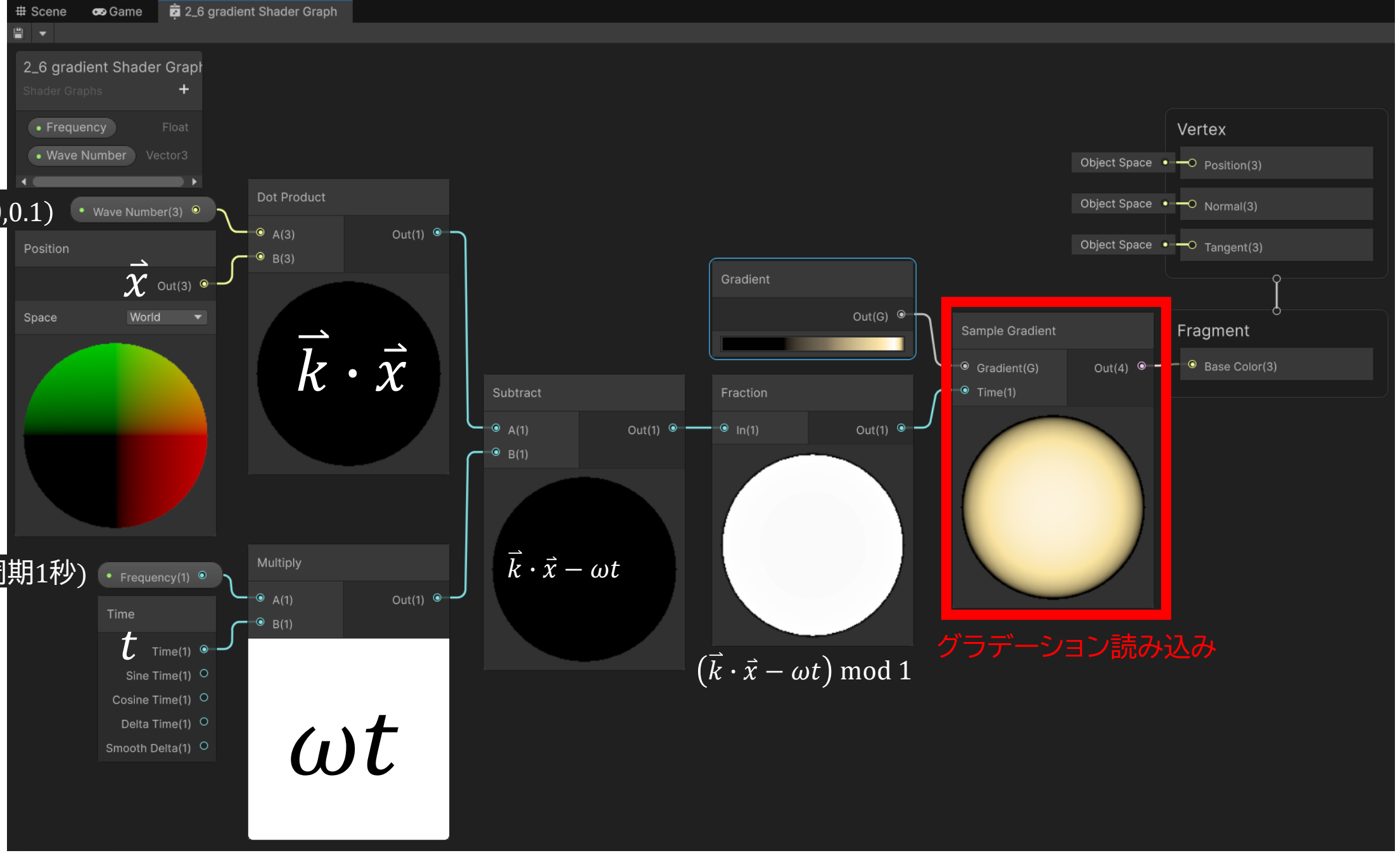
本日の内容

- 光を走らせる
  - 概要
  - 回してみよう
  - 光を走らせてみよう
    - Sin波
    - 滑らかなSin波
    - 進行波
    - のこぎり波
    - 光のデザイン
    - グラデーションの光
  - 思った場所に走らせよう

もっとデザインする

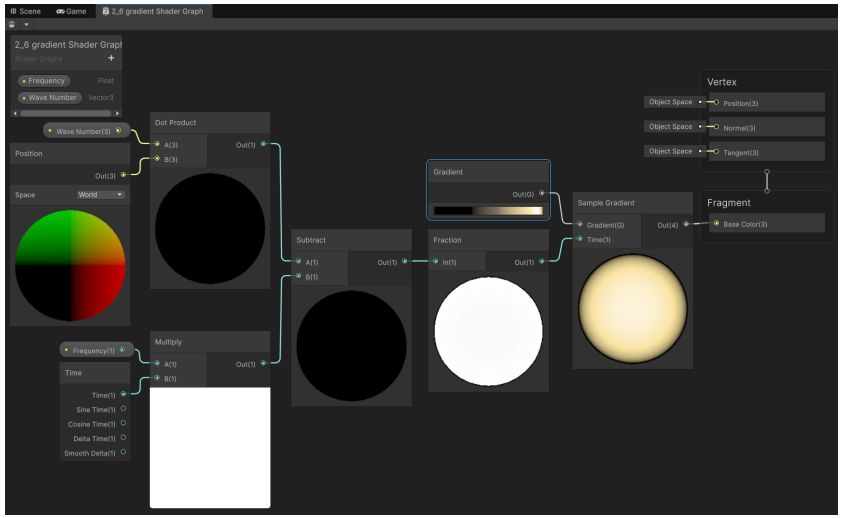
- 光るパターンを用意する
  1. Gradient Nodeを使う
  2. アーティストに光り方を制御してもらう画像を作ってもらう

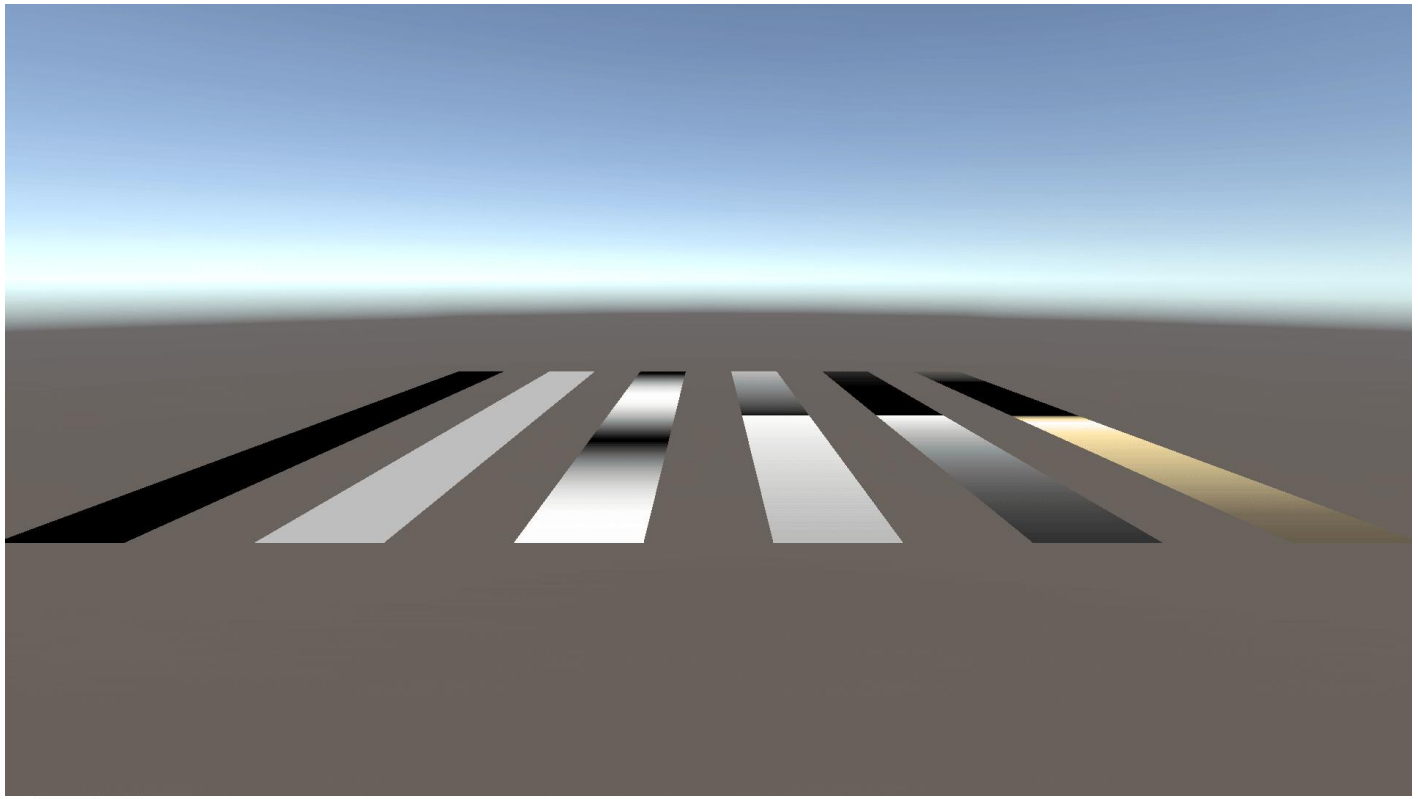



Scene
Game
2_6 gradient Shader Graph
2_6 gradient Shader Graph
Shader Graphs
Frequency
Float
Wave Number
Vector3
$\vec{k} = (0,0,0.1)$
Wave Number(3)
Position
$\vec{x}$
Out(3)
Space
World
Dot Product
A(3)
B(3)
Out(1)
$\vec{k} \cdot \vec{x}$
$\omega = 1.0$ (周期1秒)
Frequency(1)
Time
t
Time(1)
Sine Time(1)
Cosine Time(1)
Delta Time(1)
Smooth Delta(1)
Multiply
A(1)
B(1)
Out(1)
ωt
Subtract
A(1)
B(1)
Out(1)
$\vec{k} \cdot \vec{x} - \omega t$
Gradient
Out(G)
Fraction
In(1)
Out(1)
$(\vec{k} \cdot \vec{x} - \omega t) \bmod 1$
Sample Gradient
Gradient(G)
Time(1)
Out(4)
グラデーション読み込み
Vertex
Object Space
Position(3)
Object Space
Normal(3)
Object Space
Tangent(3)
Fragment
Base Color(3)

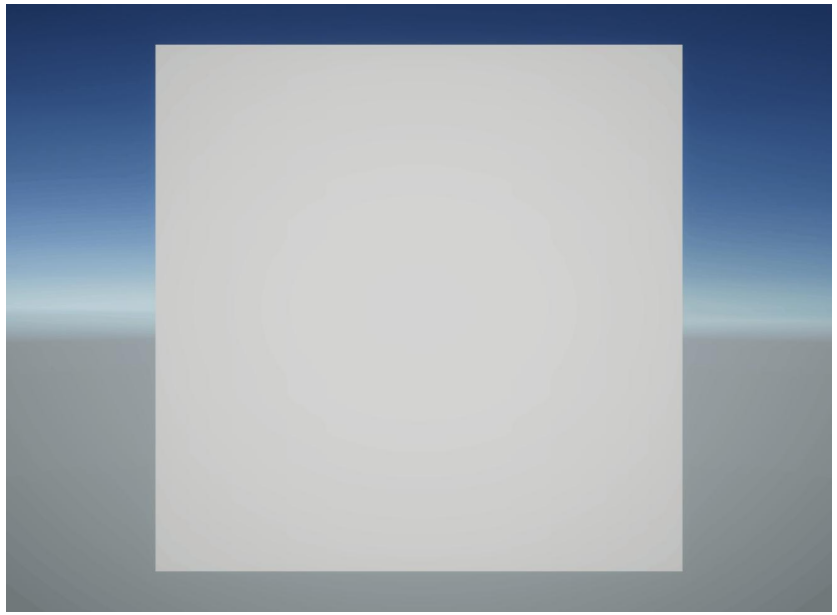
やってみよう

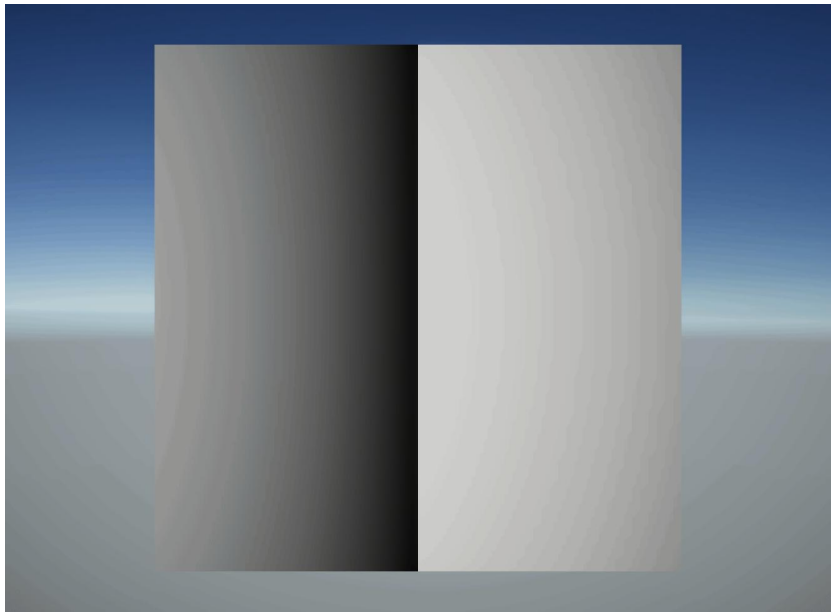
- 「2_6 gradient Shader Graph」を
  この状態にしてください



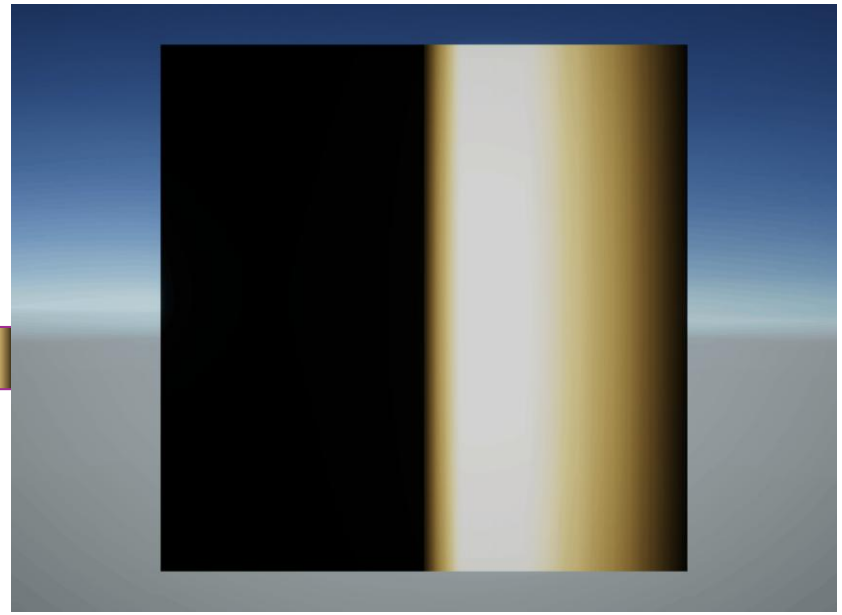


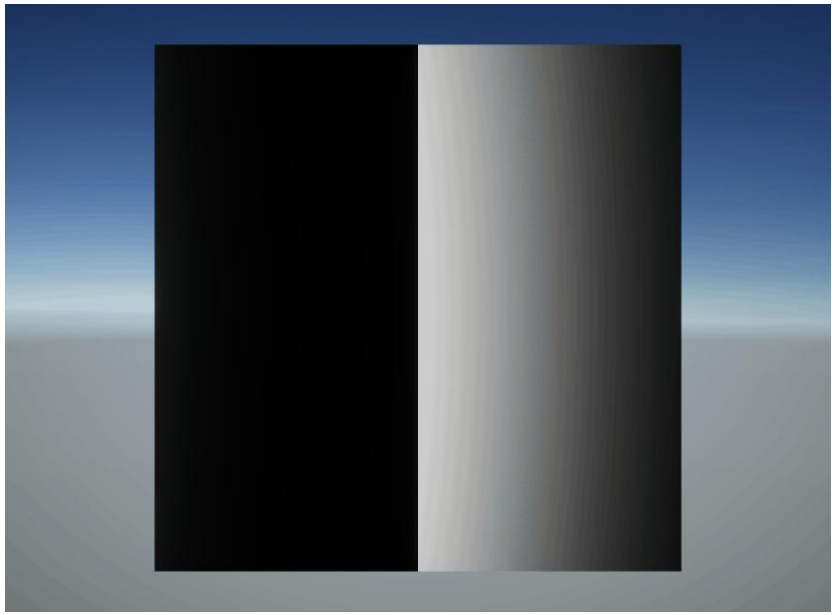
$y = \frac{1}{2} + \frac{1}{2}\cos 2\pi t$





$y = (-\omega t) \bmod 1$



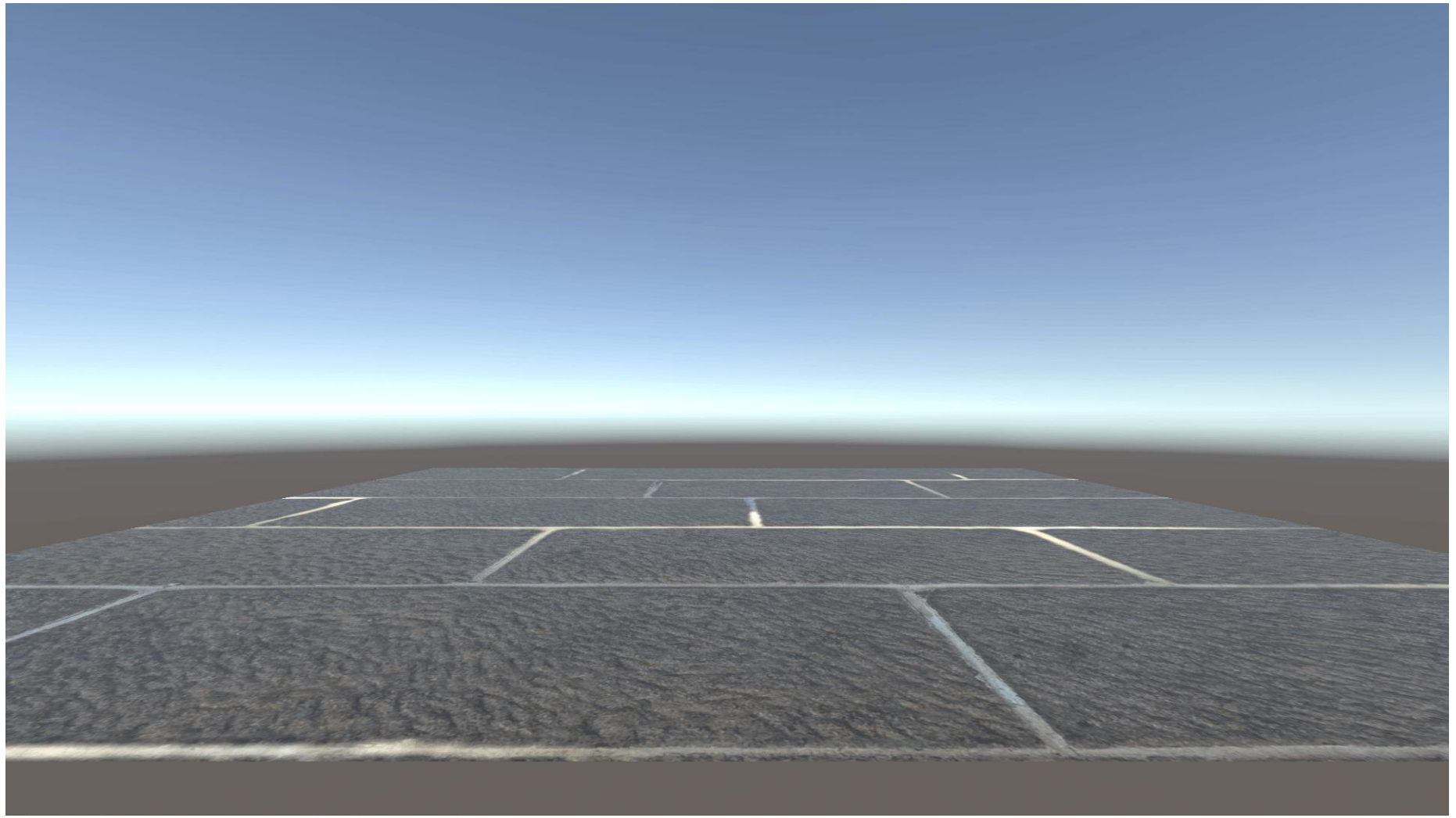


$y = \left((-\omega t) \bmod 1\right)^5$

本日の内容

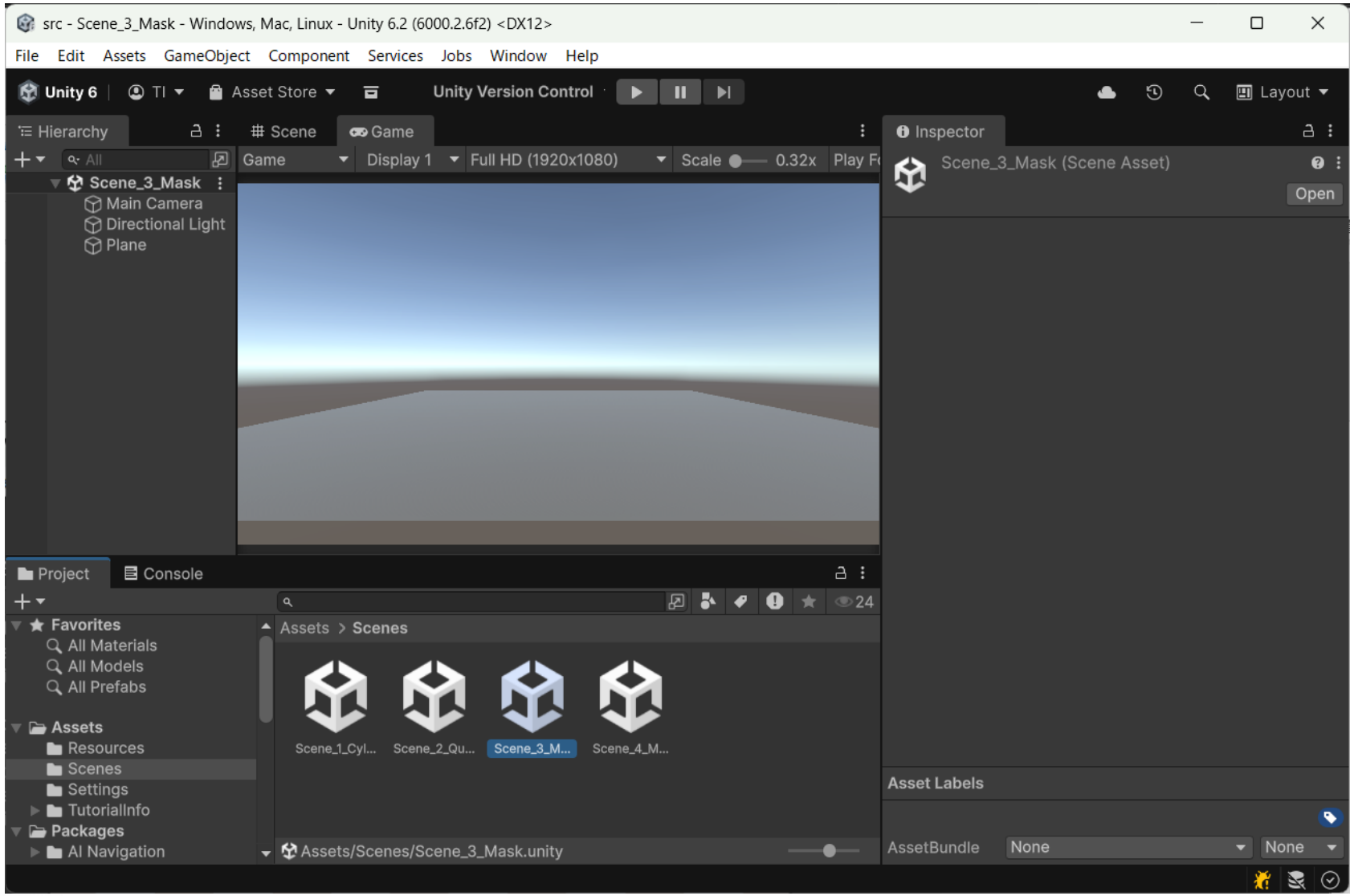
- 光を走らせる
  - 概要
  - 回してみよう
  - 光を走らせてみよう
    - Sin波
    - 滑らかなSin波
    - 進行波
    - のこぎり波
    - 光のデザイン
  - 思った場所に走らせよう

一部だけ光らせたい



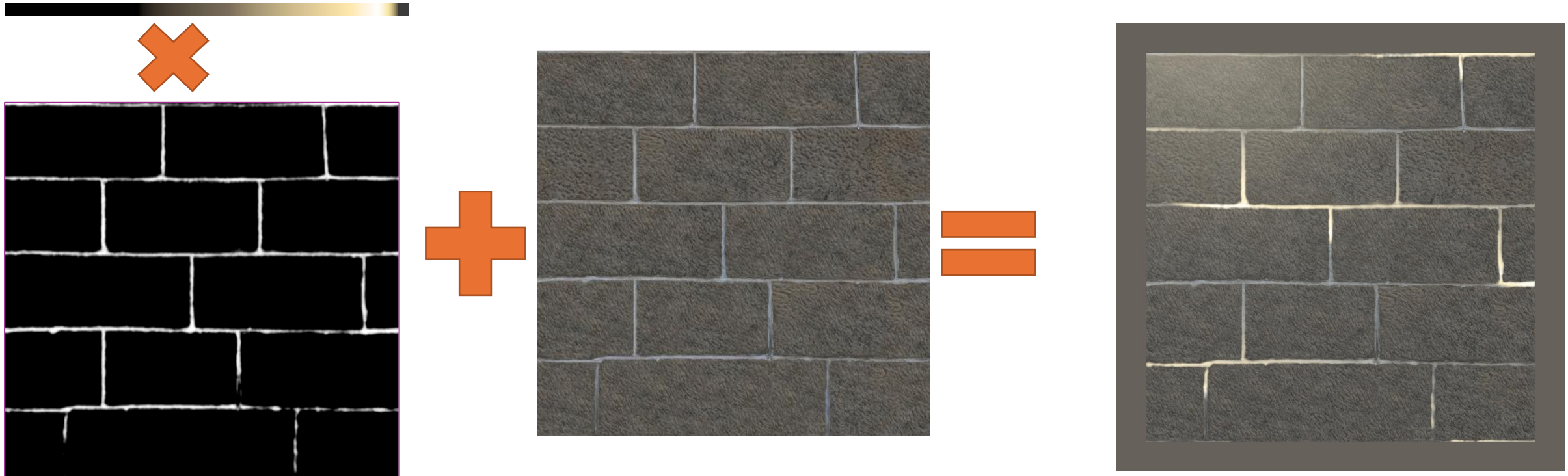
シーン

- Scene_3_Mask



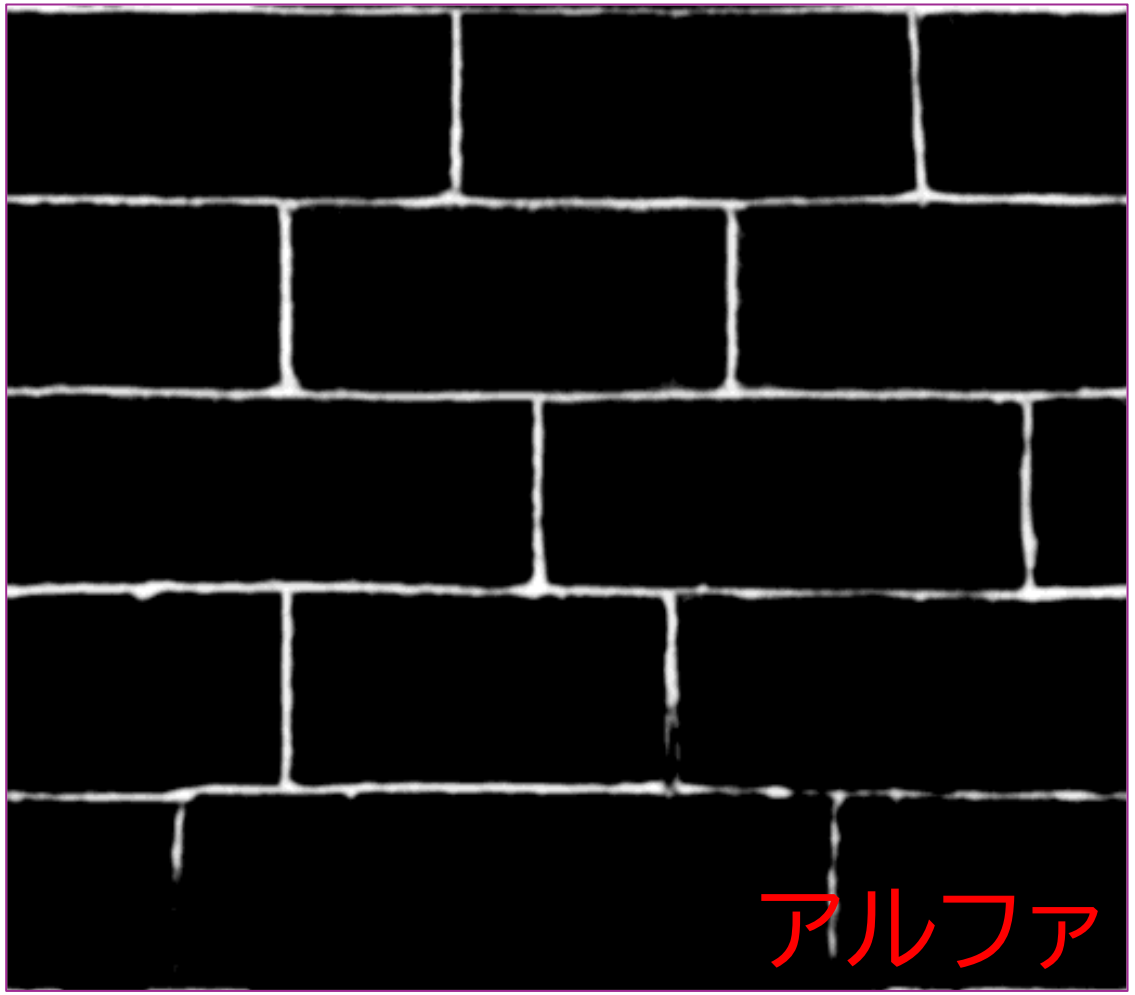
考え方

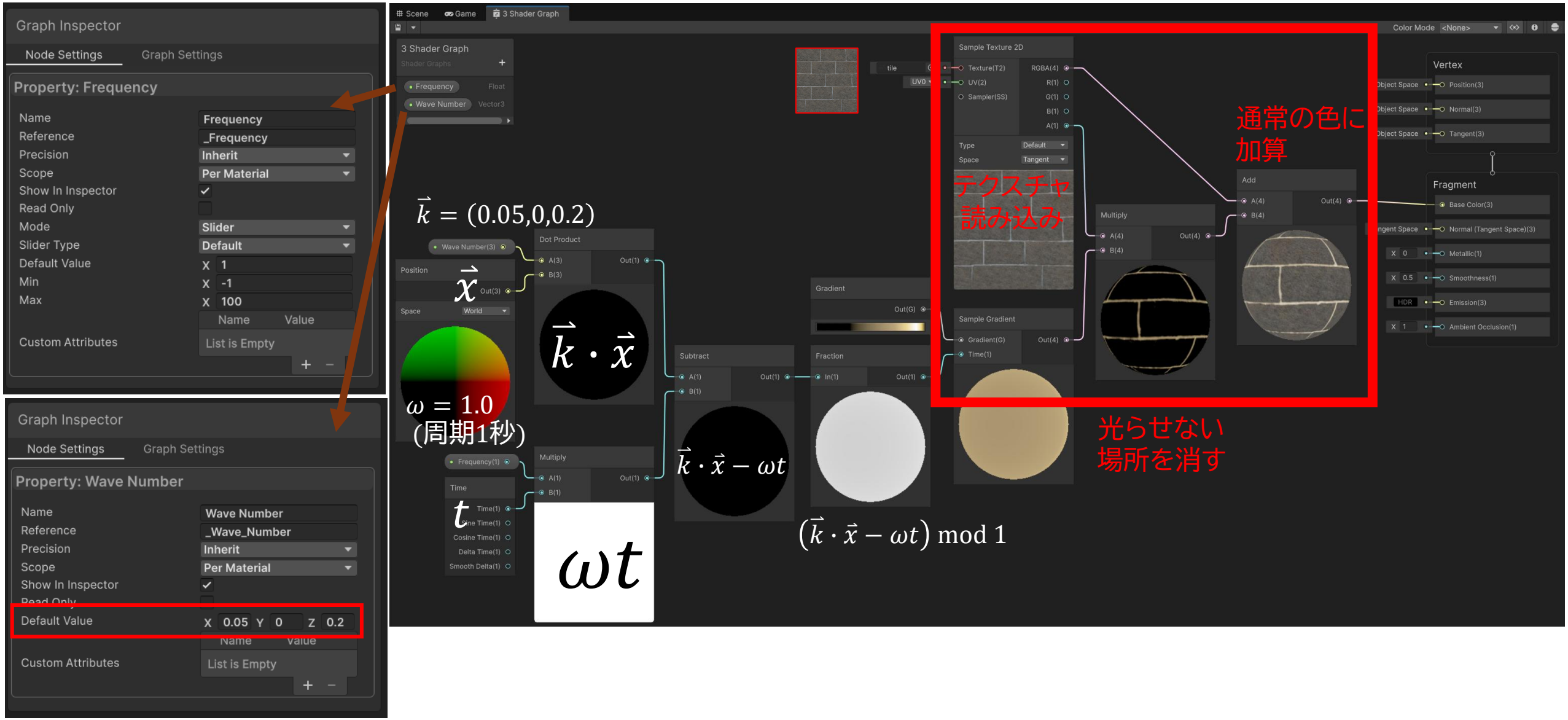
- 光らせたい部分だけ光跡を載せる
  - 光らせない部分は強さを0にする



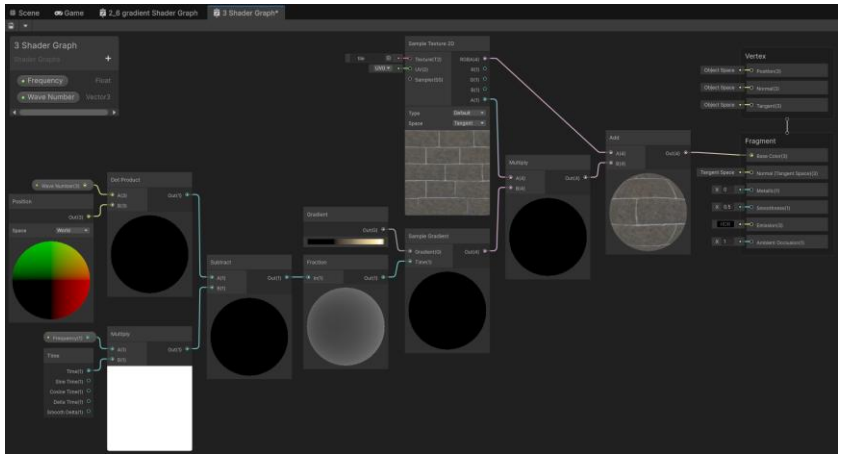
画像を用意しました

tile.png

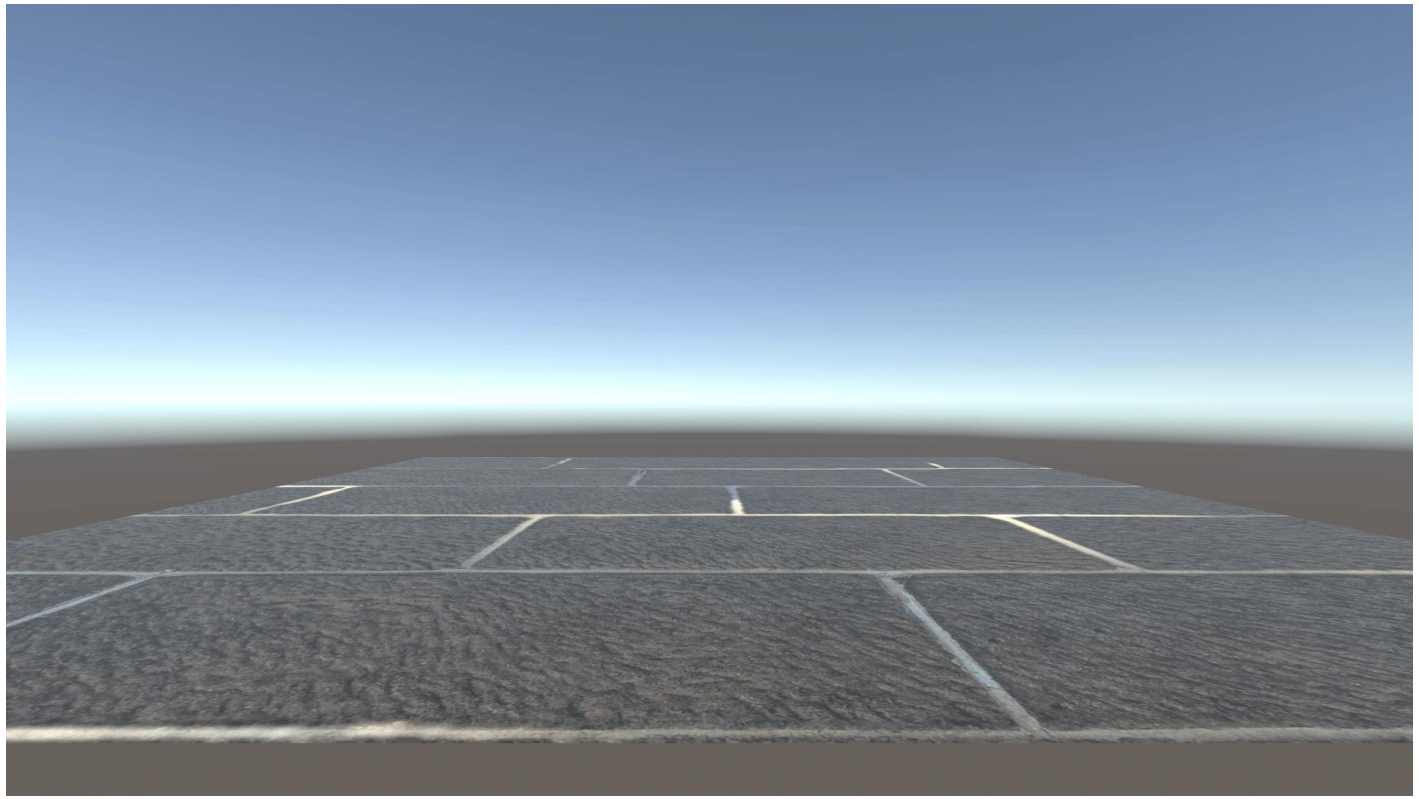



Graph Inspector
Node Settings
Graph Settings
Property: Frequency
Name
Frequency
Reference
_Frequency
Precision
Inherit
Scope
Per Material
Show In Inspector
Read Only
Mode
Slider
Slider Type
Default
Default Value
X 1
Min
X -1
Max
X 100
Name
Value
Custom Attributes
List is Empty
Graph Inspector
Node Settings
Graph Settings
Property: Wave Number
Name
Wave Number
Reference
_Wave_Number
Precision
Inherit
Scope
Per Material
Show In Inspector
Default Value
X 0.05 Y 0 Z 0.2
Custom Attributes
List is Empty
Scene
Game
3 Shader Graph
Color Mode
<None>
3 Shader Graph
Shader Graphs
Frequency
Float
Wave Number
Vector3
$\vec{k} = (0.05,0,0.2)$
$\vec{x}$
$\omega = 1.0$
(周期1秒)
t
ωt
$\vec{k} \cdot \vec{x}$
$\vec{k} \cdot \vec{x} - \omega t$
$(\vec{k} \cdot \vec{x} - \omega t) \bmod 1$
Dot Product
Subtract
Fraction
Gradient
Multiply
Time
Position
Space
World
Sample Texture 2D
Texture(T2)
UV(2)
Sampler(SS)
RGBA(4)
R(1)
G(1)
B(1)
A(1)
Type
Default
Space
Tangent
Sample Gradient
Gradient(G)
Time(1)
Add
Vertex
Position(3)
Normal(3)
Tangent(3)
Fragment
Base Color(3)
Normal (Tangent Space)(3)
Metallic(1)
Smoothness(1)
Emission(3)
Ambient Occlusion(1)
テクスチャ読み込み
通常の色に加算
光らせない場所を消す

やってみよう



- 「3 Shader Graph」を
  この状態にしてください



Tips

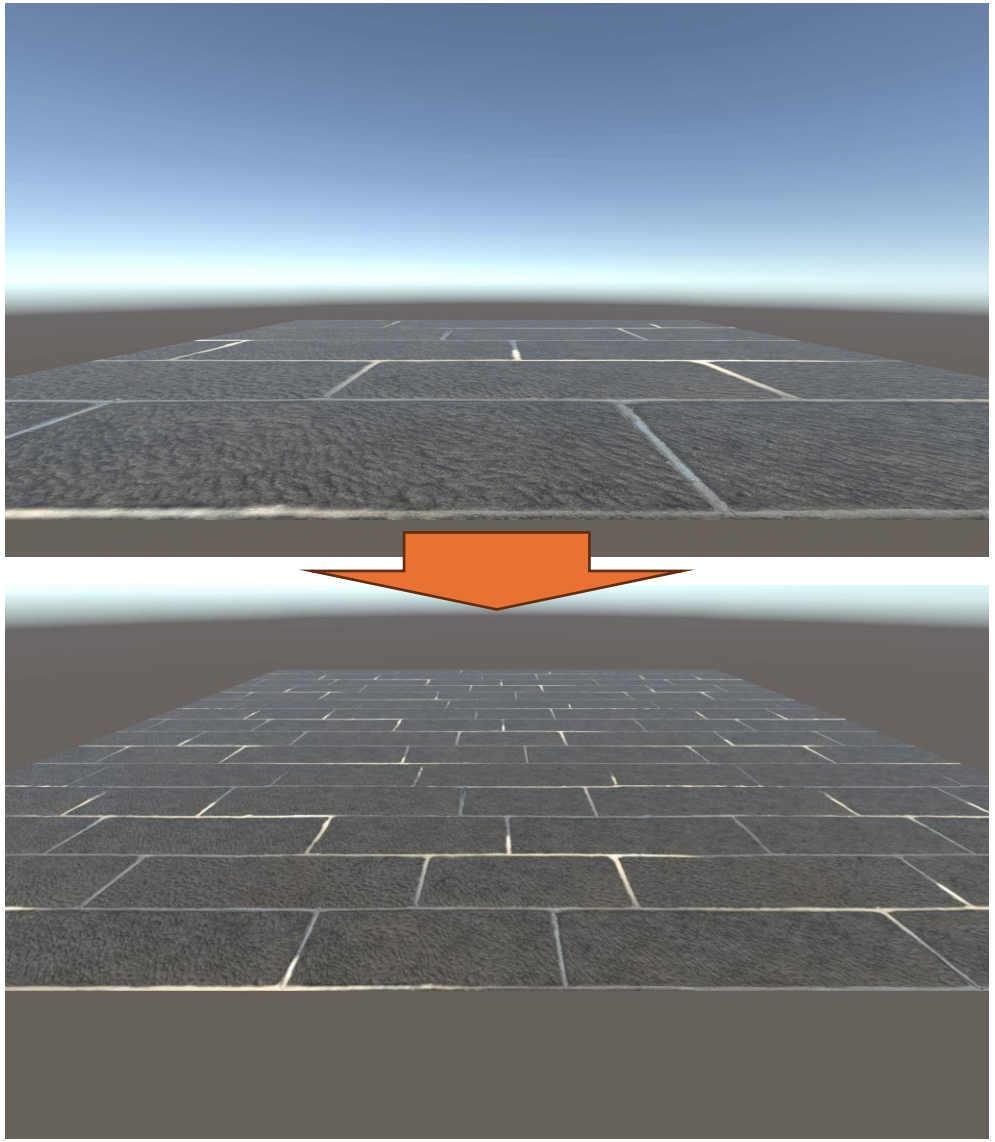
- 波数ベクトルを斜めに
  設定しているので、
  ブロックの目にそわない、
  単調でない光の動きを実現

今回やっていないこと

- 光をあみだくじのように
  経路にそって
  (一定の速度で)流すには?
  - UV情報が欲しい
    - 光が通るパターンにそって
      テクスチャ座標を並べる

補足

- テクスチャをシームレスにすると繋げても連続的に流せる
  - モデルの作り方やUVの貼り方によって、きれいに流すにはパラメータを調整する必要がある
    - 今回はワールド空間の位置座標を使っているので、必ず連続



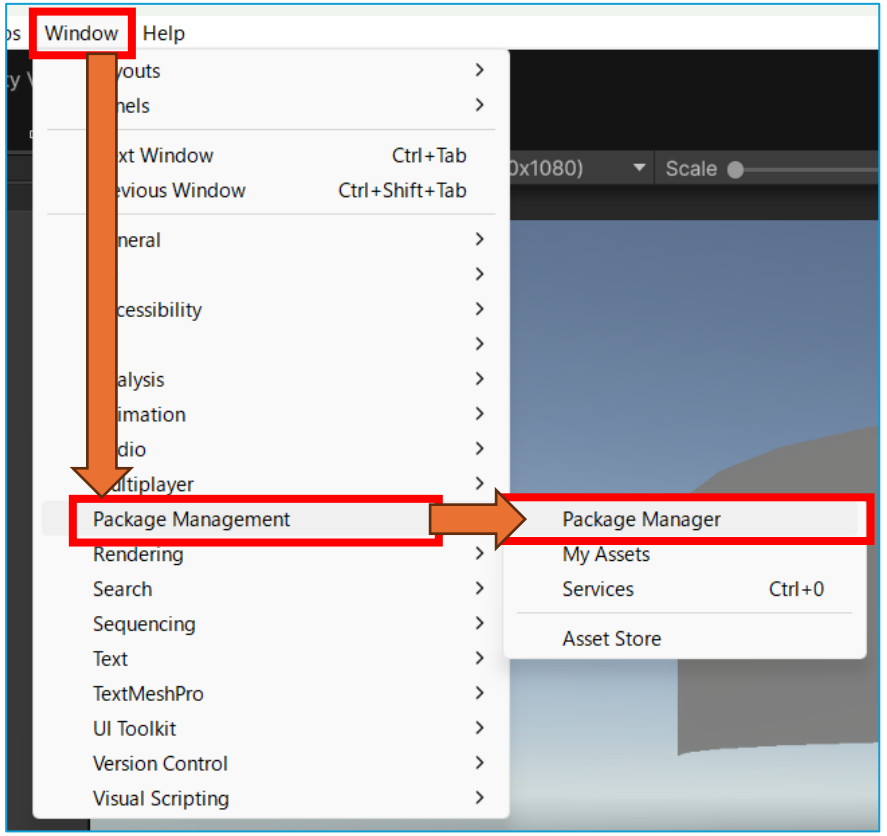
本日の内容

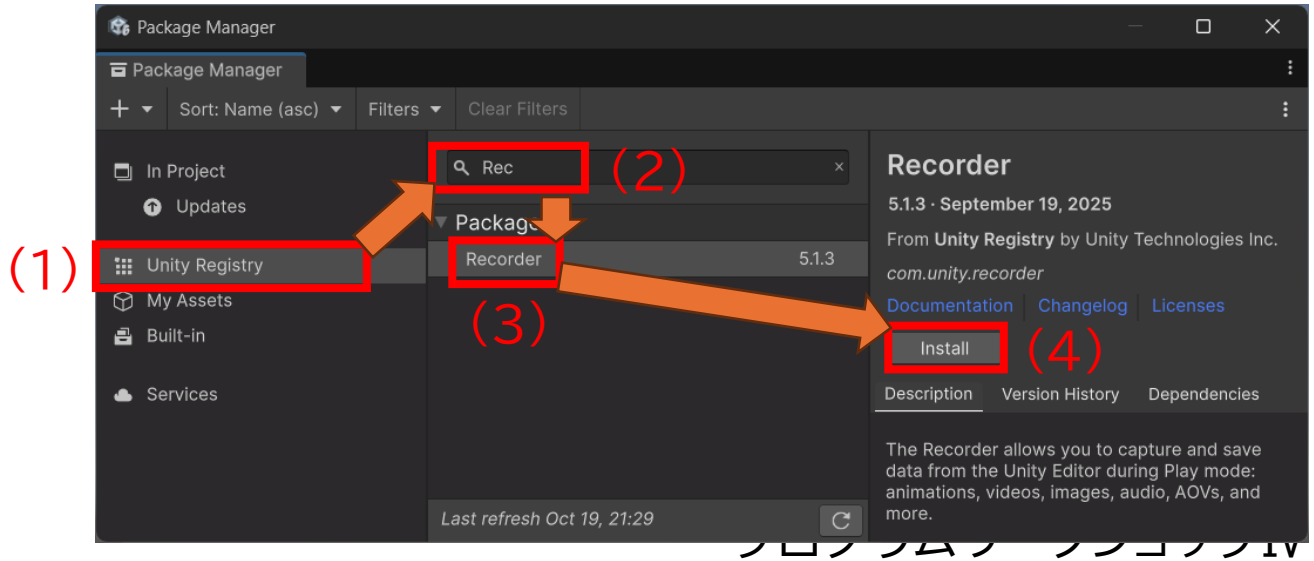
- 光を走らせる
  - 概要
  - 回してみよう
  - 光を走らせてみよう
  - 思った場所に走らせよう
- おまけ

おまけ: Unity Recorder

ゲームの動画撮影

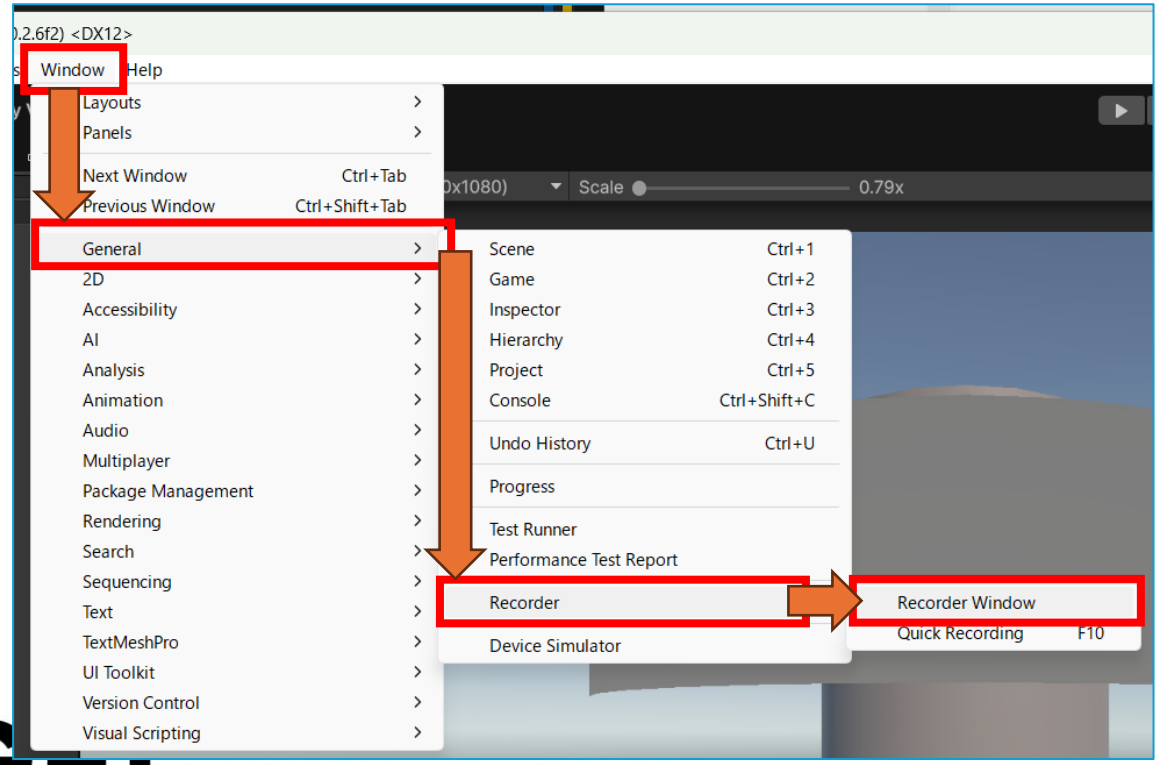
- インストール: Package Manager
  1. 「Packages」を「Unity Registry」に変更
  2. 「Rec」で検索すると、左のメニューでUnity Recorderが残る
  3. Unity Recorderを選択
  4. インストール

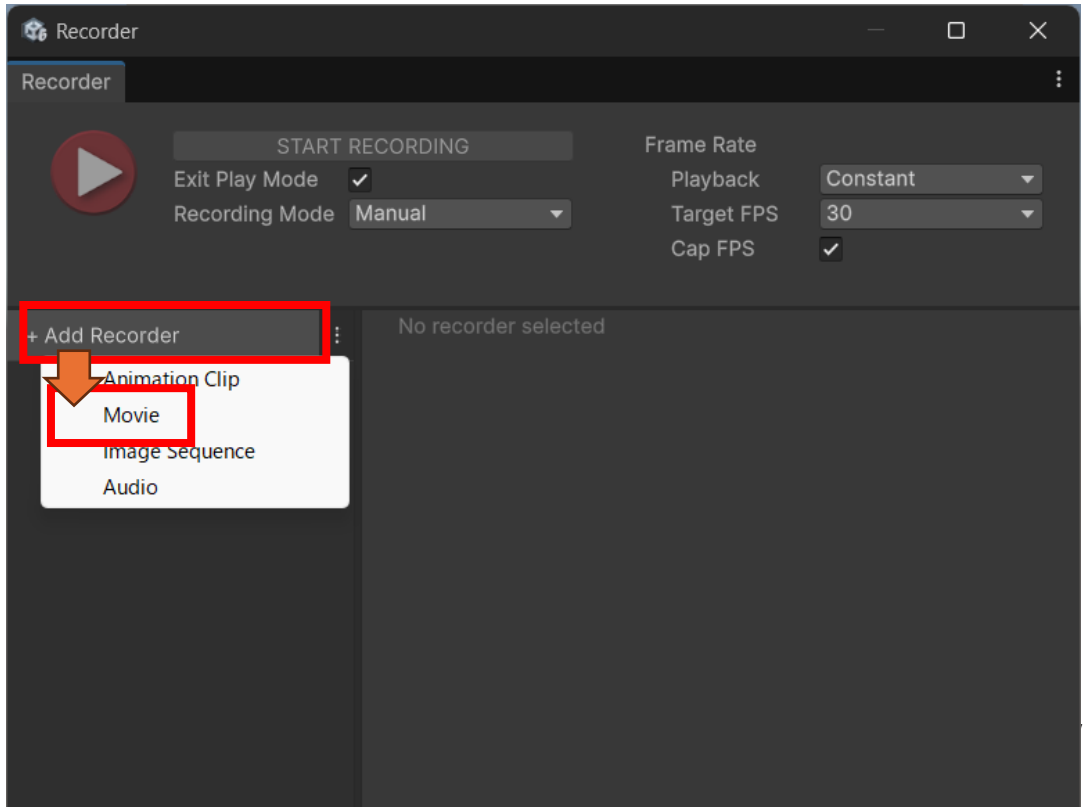


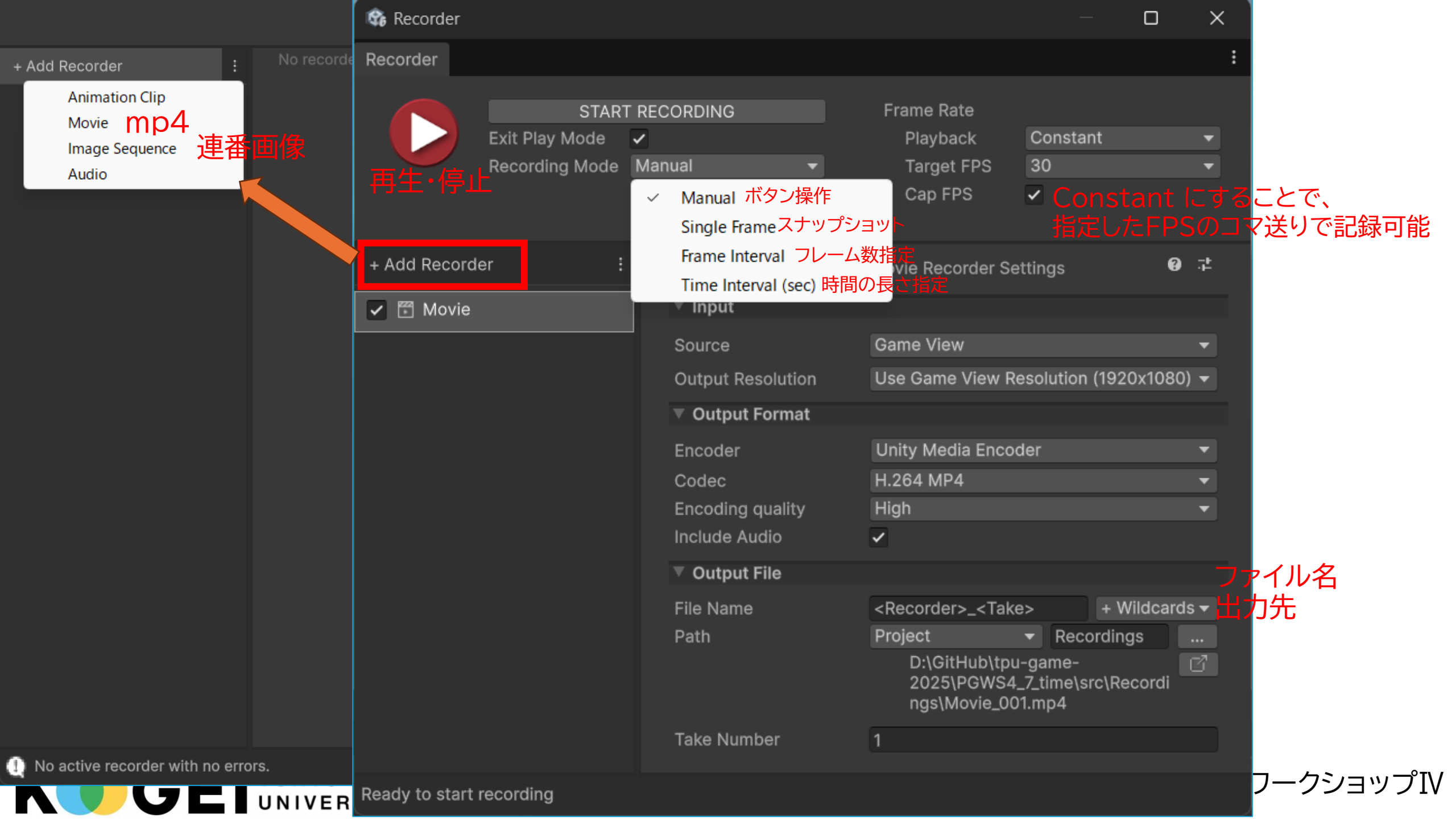


Unity Recorder の使い方

- 起動
  - Window メニュー
    - General → Recorder → Recorder Window






+ Add Recorder
Animation Clip
Movie
mp4
Image Sequence
連番画像
Audio
No recorde
No active recorder with no errors.
Recorder
Recorder
START RECORDING
再生・停止
Exit Play Mode
Recording Mode
Manual
Manual ボタン操作
Single Frame スナップショット
Frame Interval フレーム数指定
Time Interval (sec) 時間の長さ指定
Frame Rate
Playback
Constant
Target FPS
30
Cap FPS
Constant にすることで、
指定したFPSのコマ送りで記録可能
+ Add Recorder
Movie
vie Recorder Settings
Input
Source
Game View
Output Resolution
Use Game View Resolution (1920x1080)
Output Format
Encoder
Unity Media Encoder
Codec
H.264 MP4
Encoding quality
High
Include Audio
Output File
File Name
<Recorder>_<Take>
+ Wildcards
Path
Project
Recordings
...
D:\GitHub\tpu-game-2025\PGWS4_7_time\src\Recordings\Movie_001.mp4
ファイル名
出力先
Take Number
1
Ready to start recording

Recorder

Recorder

START RECORDING

Exit Play Mode ✓

Recording Mode Frame Interval

Start 0 End 30

Frame Rate

Playback Constant

Target FPS 30

Cap FPS ✓

30フレ(target FPSが30なので1秒)で終了

+ Add Recorder

Movie

Recorder Type Movie Recorder Settings

Input

Source Game View

Output Resolution Use Game View Resolution (1920x1080)

Output Format

Encoder GIF Encoder

Quality 90

Loop ✓

Output File

GIFアニメーションの出力も可能

File Name <Recorder>_<Take> + Wildcards

Path Project Recordings ...

D:\GitHub\tpu-game-2025\PGWS4_7_time\src\Recordings\Movie_003.gif

Take Number 3

Ready to start recording

Recordings

新規作成 プレビュー

PGWS4_7_time

.git

src

.vs

Assets

Library

Logs

obj

Packages

ProjectSettings

Recordings

Temp

UserSettings

Movie_001.gif

1 個の項目

.gitignoreにディレクトリを追加しよう

- 最新の.gitignoreでは追加されている

src > .gitignore

```
1   # This .gitignore file should be placed at the root of your Un
2   #
3   # Get latest from https://github.com/github/gitignore/blob/mai
4   #
5   .utmp/
6   /[Ll]ibrary/
7   /[Tt]emp/
8   /[Oo]bj/
9   /[Bb]uild/
10  /[Bb]uilds/
11  /[Ll]ogs/
12  /[Uu]ser[Ss]ettings/
13  /[Rr]ecordings/
14  *.log
```

まとめ

- Timeノードを使うとシェーダだけで動かせる
  - Unityの実行時間で動く
  - 「Time」出力は1秒ごとに1.0増える
    - 長時間実行すると、精度が下がるので注意
  - 「Sine Time」,「Cosine Time」出力は2π(≈6.28)秒で元に戻る
    - 周期を変えることはできない
- マスクを用いたシェーダの切り分け
  - アルファ成分などを適用する重みに使う