UnityプロジェクトをGitで管理

#個人開発集会 2024.10.24 HIDE

今日話すこと

- UnityのアバターやワールドのプロジェクトでGitを使う話

自己紹介

- HIDE
  - データサイエンス、ソフトウェアの仕事してます
  - ハードウェアもわかります
  - VRChat: hide3791
  - X: @NPComplete00

UnityでGitを
やってみようと思ったきっかけ

- アバター改変とかしてると、プロジェクトがよく分からない状態になる
- 記憶力がないので、少し時間が空くと何をどこまでやっていたか忘れる
- アバターが壊れて、今日のイベントに着ていくアバターがない状況に...

Gitを使うことで期待していたこと

- 自分がやった作業を記録できる
- きちんと動いていた、あのときに戻せる
- いろいろな変更を同時に進められる

さっそくGitを導入...その前に

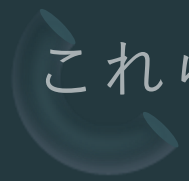
- Unityプロジェクトはデータ量が大きい
- 全ファイルをGitの管理下に入れるのは効率よくない

まずは、Unityプロジェクトの構成を確認しないと、、、

Unityプロジェクトのフォルダ構成

あるアバターのプロジェクトのデータ量は1.5GB、けっこう大きい

```
$ du -mh ./ --max-depth=1
8.0M    ./Assets
1.4G    ./Library
161K    ./Logs
81M     ./Packages
111K    ./ProjectSettings
41K     ./UserSettings
```



これらすべてをGitの管理下に入れる必要があるはずがない、と思った

Gitで管理する/しないファイル

ありがたいことに、Unityプロジェクト向けの.gitignoreが公開されている

https://github.com/github/gitignore/blob/main/Unity.gitignore

結論としては、

- Library、Logs、UserSettingsは管理不要
- Assets、Packages、ProjectSettingsは一部を管理

```
$ du -mh ./ --max-depth=1
8.0M    ./Assets
~~1.4G    ./Library~~
~~161K    ./Logs~~
81M     ./Packages
111K    ./ProjectSettings
~~41K     ./UserSettings~~
```

バイナリデータの管理

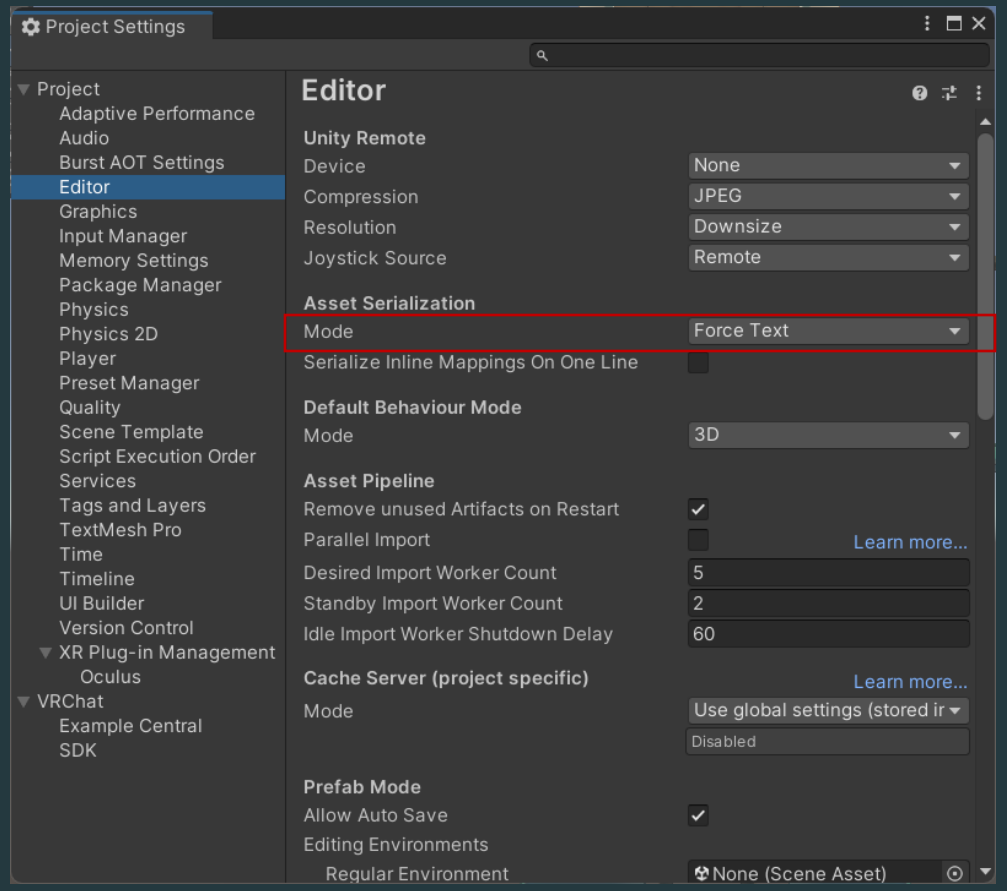
- 絵や音声といったバイナリデータはGit LFSで管理する
  - *.png、*.jpg、*.fbx、*.dll、*.dylib、*.exe、*.wav、*.mp3

環境

- Windows 11
- Unity 2022.3.22f1
- VRChat Creator Companion v2.3.4
- Git for Windows 2.46.0
- Git Bash

Gitを使うためのUnityの設定

- ForceText: すべてのアセットをテキストモードに変換する



実際にやってみる
「アバターをアップロードするまで」

1. アバターのプロジェクトを作る
2. アバターのパッケージをインポートする
3. ビルド＆アップロードする

てっさくちゃん を使わせていただきました

1. アバターのプロジェクトを作る

- アバターのプロジェクトを作って、SDKとShaderをインポートする

```
$ vpm new Tessakuchan Avatar
$ vpm add package com.vrchat.base -p Tessakuchan
$ vpm add package com.vrchat.avatars -p Tessakuchan
$ vpm add package jp.lilxyzw.liltoon -p Tessakuchan
```

- Gitで管理しないファイル、LFSで管理するファイルを設定して、コミット

```
$ cd Tessakuchan
$ cp somewhere/Unity.gitignore .gitignore
$ git init .
$ git lfs install
$ git lfs track "*.png" "*.jpg" "*.fbx" "*.dll" "*.dylib" "*.exe" "*.wav" "*.mp3"
$ git add .gitignore .gitattributes Assets Packages ProjectSettings
$ git commit -m "Initial"
```

2. アバターのパッケージをインポートする

- シーンファイルを作って、コミットする

```
$ git add Assets Packages ProjectSettings
$ git commit -m "Make scene file."
```

- Tessakuchan_ver1.0.5.unitypackageをインポートして、コミットする

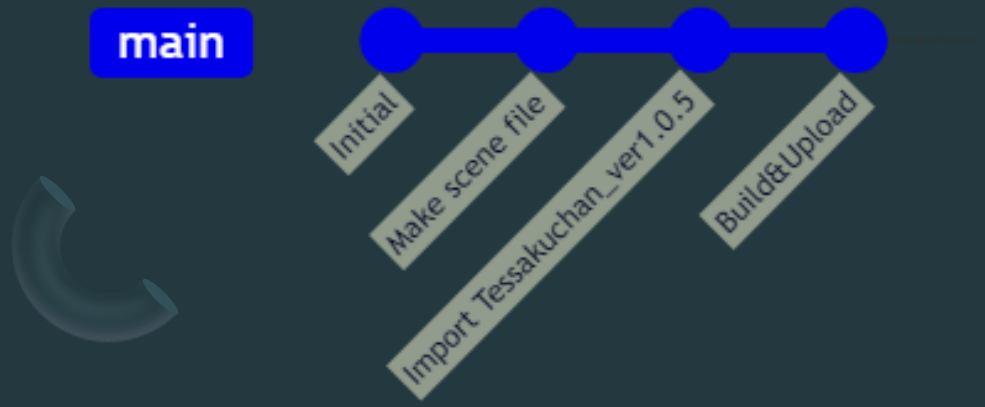
```
$ git add Assets
$ git commit -m "Import Tessakuchan_ver1.0.5"
```

3. ビルド＆アップロードする

- Build&Uploadを実行して、コミットする

```
$ git add Assets ProjectSettings
$ git commit -m "Build&Upload"
```

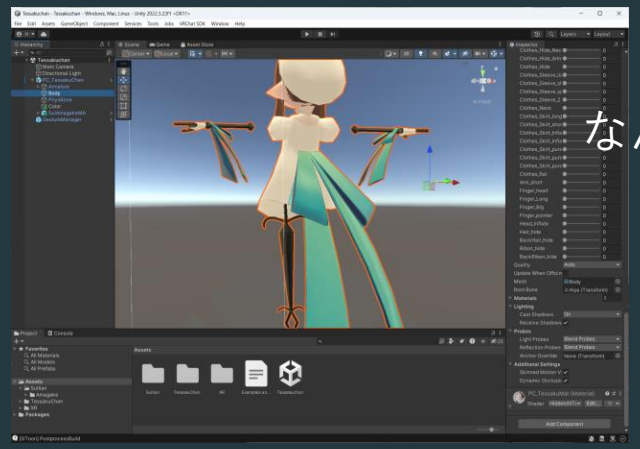
- Commit Treeはこうなっている



実際にやってみる
「改変しているときによくある操作」

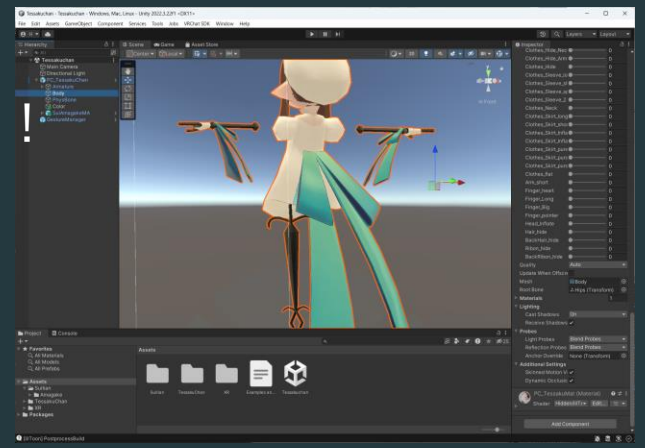
- 設定変えてみたけど、よくわからなくなったので元に戻したい
- 服の色とかをいろいろ変えてみて、気に入ったのを選びたい

設定変えてみたけど、
よくわからなくなったので元に戻したい



なんか変わった！

元に戻せた！



```
$ git status
On branch main
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   Assets/Tessakuchan.unity

$ git checkout Assets/Tessakuchan.unity    ← 最新コミットの状態に戻る
Updated 1 path from the index
```

服の色とかをいろいろ変えてみて、気に入ったのを選びたい

例として、マテリアルを変える場合で説明します

1. ブランチを作って、そのブランチに切り替える

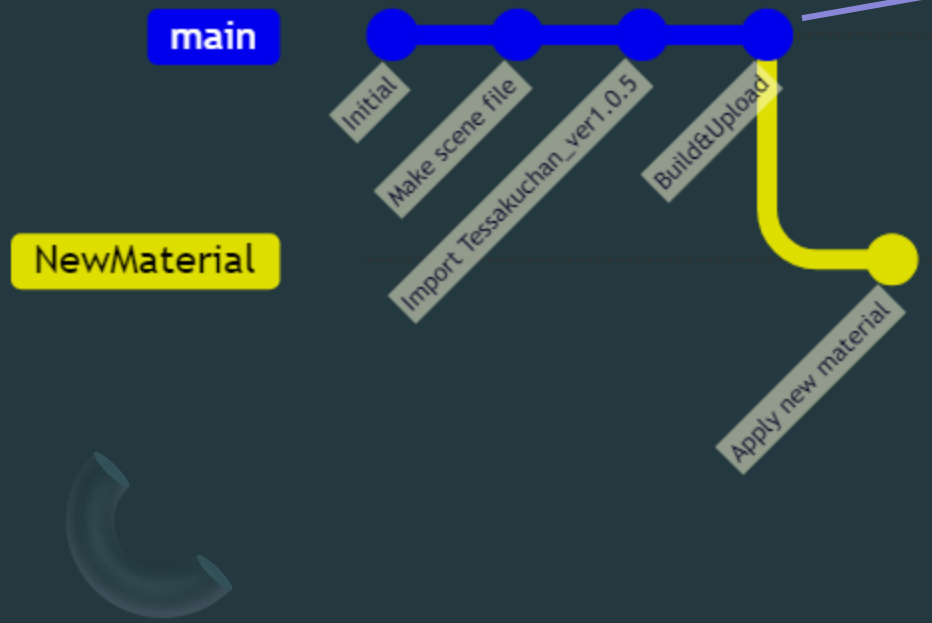
```
$ git checkout -b NewMaterial
```

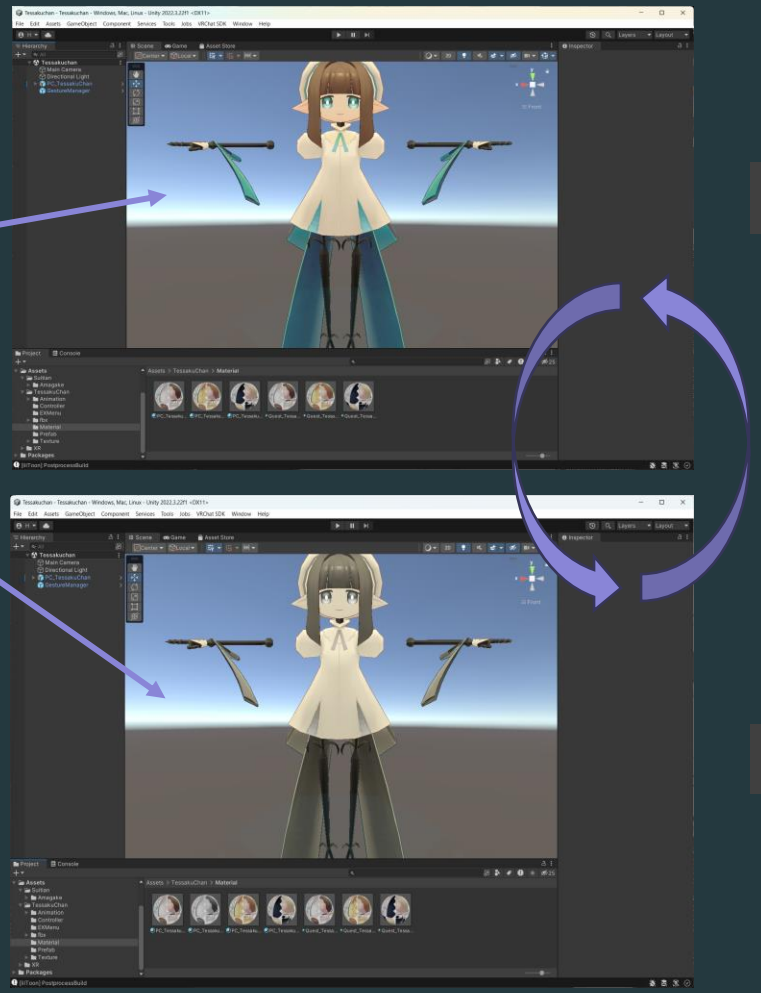
2. 色を変更したテクスチャファイルをプロジェクトに入れる
3. そのテクスチャを適用したマテリアルを作る
4. 対象とするメッシュに作ったマテリアルを適用する
5. 追加されたファイル、変更されたファイルをコミットする

```
$ git add Assets
$ git commit -m "Apply new material"
```

服の色とかをいろいろ変えてみて、気に入ったのを選びたい（cont'd）

グレースケールのテクスチャで
マテリアルを作って適用した





```
$ git checkout main
```

ブランチを切り替えると、
即座に変わる

```
$ git checkout NewMaterial
```

服の色とかをいろいろ変えてみて、気に入ったのを選びたい（cont'd）

```
$ git checkout -b NewMaterial
```

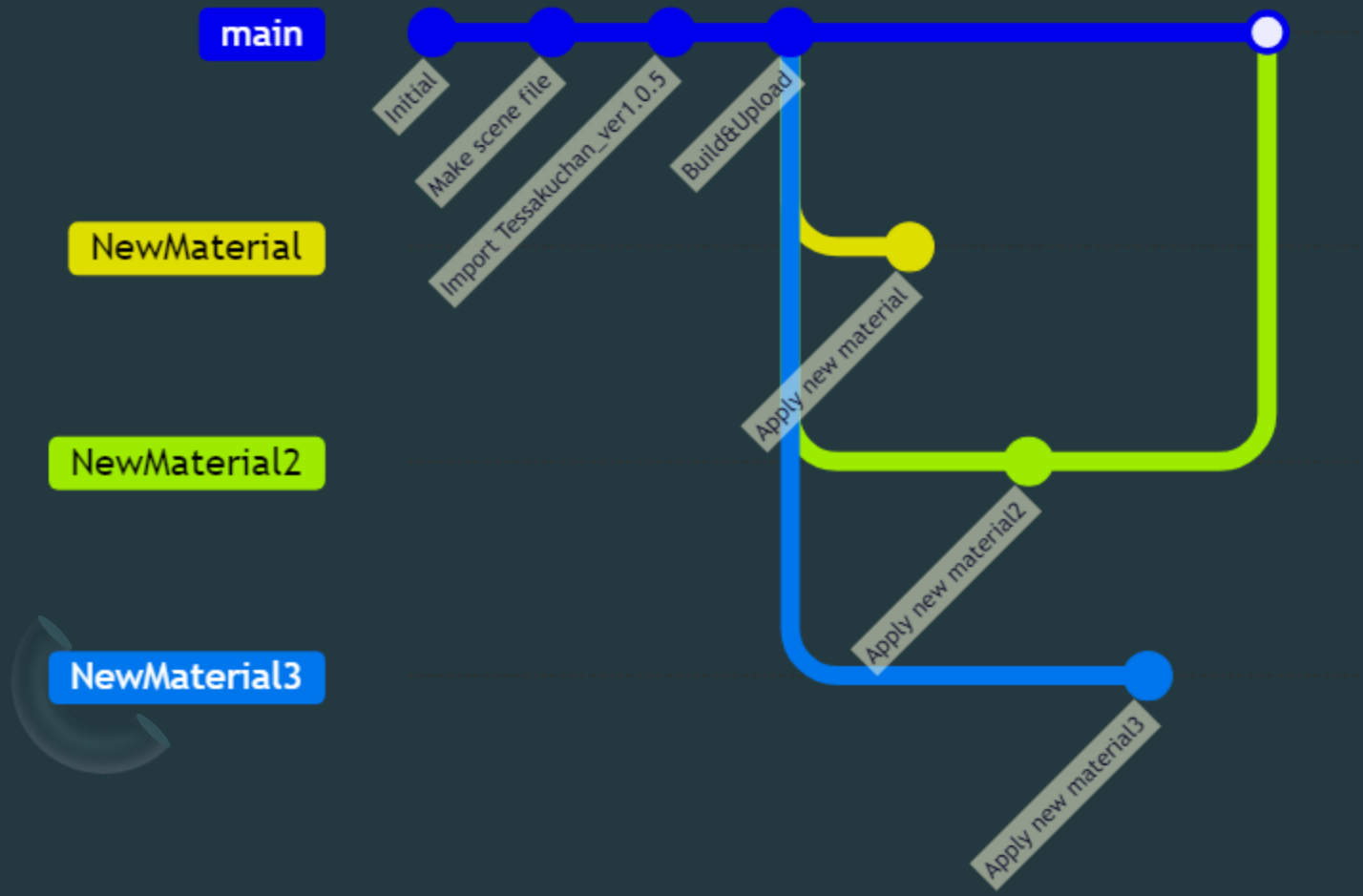
テクスチャ、マテリアルなどをUnityで操作

```
$ git status
On branch NewMaterial
Changes not staged for commit:
(use "git add ..." to update what will be committed)
(use "git restore ..." to discard changes in working directory)
      modified: Assets/Tessakuchan.unity

Untracked files: (use "git add ..." to include in what will be committed)
      Assets/TessakuChan/Material/PC_TessakuMat_Gray.mat
      Assets/TessakuChan/Material/PC_TessakuMat_Gray.mat.meta
      Assets/TessakuChan/Texture/TessakuTexGray.png
      Assets/TessakuChan/Texture/TessakuTexGray.png.meta
$ git add Assets
$ git commit -m "Apply new material"
```

シーンファイル以外に、テクスチャやマテリアルのファイルが変更されたことをGitが認識している

服の色とかをいろいろ変えてみて、気に入ったのを選びたい（cont'd）

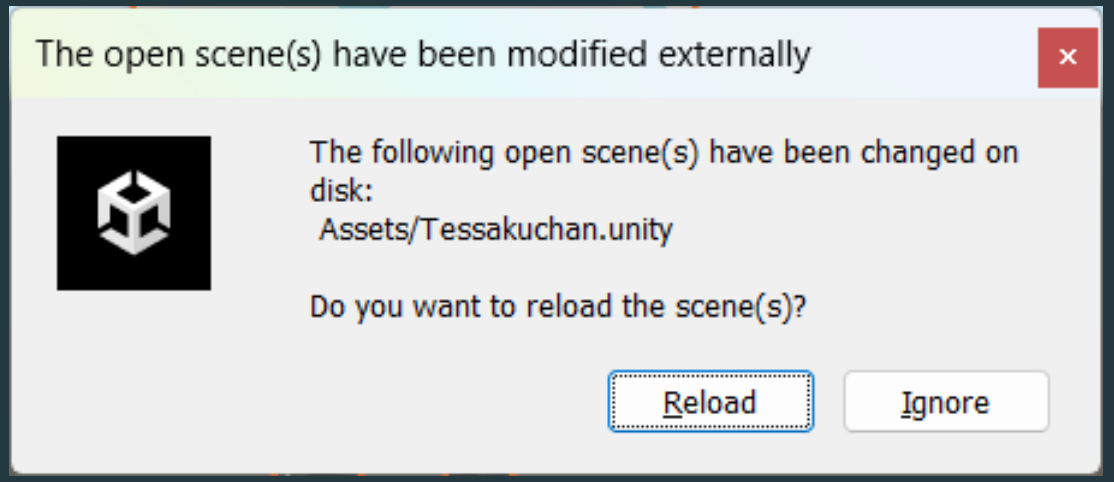


３つマテリアルを作ってみて、２つ目を使うことにした

```
$ git checkout main
$ git merge NewMaterial2
```

Unityプロジェクトのファイルを Unity外からさわった場合にどうなるのか？

- Unityはファイルの変更を見張っているので、変更されるとダイアログを表示する → 「Reload」を押すと、変更が反映される。



Gitが管理しているファイルのサイズはどれくらい？

1.5GB

プロジェクト全ファイルのサイズ

```
$ du -mh ./ --max-depth=1
8.0M    ./Assets
1.4G    ./Library
161K    ./Logs
81M     ./Packages
111K    ./ProjectSettings
41K     ./UserSettings
```

10MB

Gitが管理しているファイルのサイズ

```
$ du -mh ./ --max-depth=1
8.0M    ./Assets



960K    ./Packages
111K    ./ProjectSettings
```

※リモートリポジトリからCloneした直後を計測

おわり

- 自分がやった作業を記録できる
- きちんと動いていた、あのときに戻せる
- いろいろな変更を同時に進められる

ができることで、躊躇なくいろんなこと気軽に試せるので、Unityの上達に役立ってると思う

もっと良いやりかたがあったら是非教えてください！

UnityプロジェクトをGitで管理

Appendix
リモートリポジトリを使う

リモート(GitLab)にPushして、別のPCからCloneする場合の手順

1. GitLabでプロジェクトを作成する（Project nameはtessakuchan）
2. リモートリポジトリを設定する

Pathは実際の環境に合わせて読み替えてください

```
$ git remote add origin ssh://git@192.168.10.100:2224/hw/tessakuchan.git
```

3. リモートにPushする

```
$ git push --set-upstream origin main
```

4. 別のPCからCloneする

```
$ git clone ssh://git@192.168.10.100:2224/hw/tessakuchan.git
```