☒Unity × GitHub × Visual Studio(ChatGPT様Says)

☒ 目的

Unity プロジェクトを GitHub で安全に管理し、Visual Studio から快適に操作するための手順をまとめました。
この資料を見れば、**チーム制作でも壊れないUnityリポジトリ構成**が理解できます。

☒ 1. Unity 側の設定

① Editor 設定を変更する

メニューから
「Edit → Project Settings → Editor」 を開きます。

設定項目	値
Version Control	**Visible Meta Files**
Asset Serialization	**Force Text**

これにより、ファイルごとの変更がテキスト化され、Gitで差分が取れるようになります。

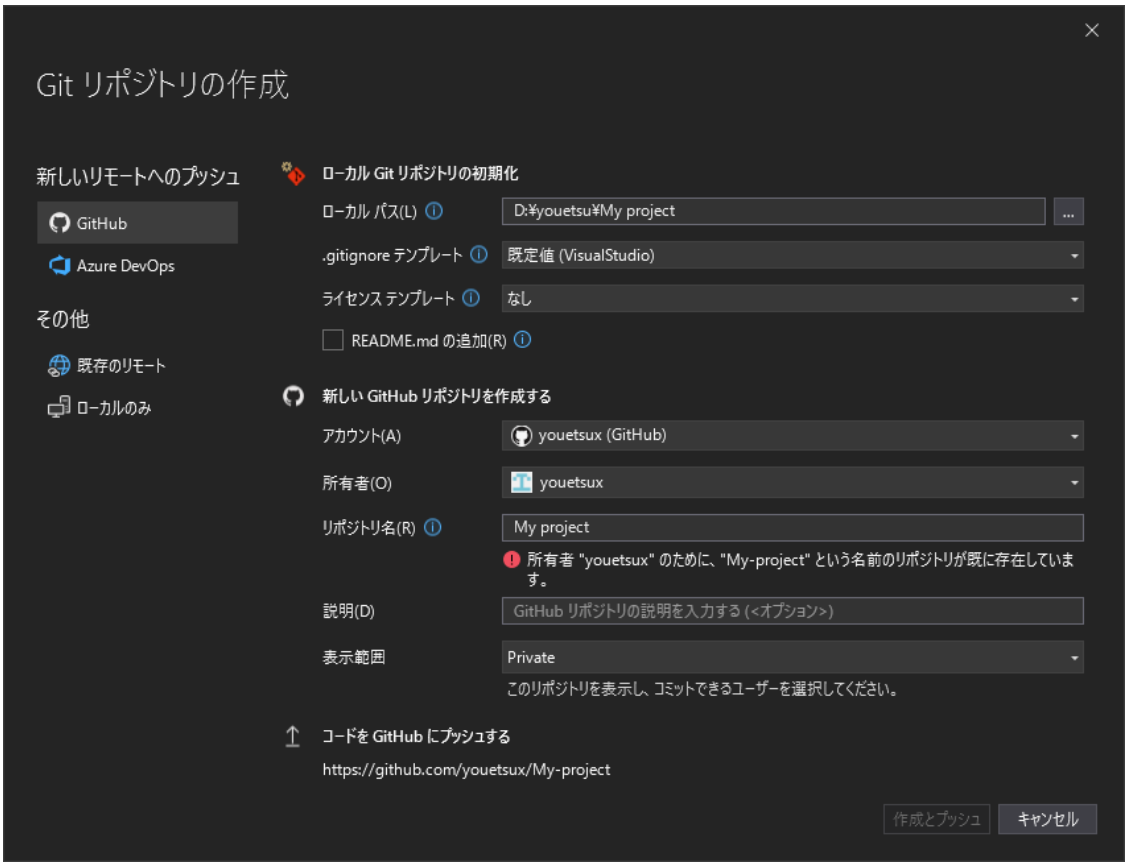
---

② Unity バージョンをそろえる

全員が同じ **Unity LTS（例：2022.3系）** を使うこと。
`ProjectSettings/ProjectVersion.txt` が一致しているか確認しましょう。

☒ 2. .gitignore の設定

Visual Studio の「Git」メニューから
[.gitignore を追加] → Unity テンプレート選択。



.gitignoreテンプレートからUnity探して選ぶ！

その後、以下を確認または追記します。（これは自分で、プロジェクトフォルダの.gitignoreを開いて編集

```
[Ll]ibrary/
[Tt]emp/
[Oo]bj/
[Bb]uild*/
[Bb]uilds/
[Ll]ogs/
UserSettings/
*.csproj
*.sln
*.user
.vs/
```

✔☒ ポイント

- `Library/` はGitに入れない（巨大・自動再生成される）
- `.meta` ファイルは削除禁止（参照リンク切れ防止）

なんか、長いバージョンもあったよ。

.gitignore（Library を絶対入れない）

Unity公式テンプレをベースに。最低限これでOK：

```
# Unity build cache
[Ll]ibrary/
[Tt]emp/
[Oo]bj/
[Bb]uild*/
[Bb]uilds/
[Ll]ogs/
[Mm]emoryCaptures/

# User-specific
[Ss]ysinfo.txt
UserSettings/
```

```
*.csproj
*.unityproj
*.sln
*.suo
*.tmp
*.user
*.userprefs
*.pidb
*.booproj
*.svd
*.pdb

# Autogenerated
*.apk
*.aab
*.app
*.ipa

# Visual Studio cache
.vs/
```

☒ 3. Visual Studio での Git 操作

① 新しいリポジトリを作成

Git → 新しいリポジトリの作成

---

② コミットとプッシュ

Git Changes ウィンドウで変更内容を確認し、
メッセージを入力して **[コミットしてプッシュ]**

---

③ ブランチの作成と切り替え

Git Branches ウィンドウから操作可能。
例：`feature/player-move` や `fix/audio-lag` など。

---

④ リモートを登録する

Git → **リモートの管理** → GitHub のURLを追加

☒ 4. Git LFS（大容量ファイル管理）

① LFSとは？

Unityの画像・音声・FBX・動画ファイルなどをGitHubの100MB制限に引っかけずに保存する仕組みです。

自分のリポジトリのフォルダをコマンドプロンプトで開いて、以下のコマンドでgit-lfsを有効化
なんか言われたら全部Yes！

```
winget install --id GitHub.GitLFS -e
```

② 設定手順（最初の1回だけ）

Visual Studio の **「表示 → ターミナル」** を開き、以下を入力：

```
git lfs install
git lfs track "*.psd" "*.png" "*.jpg" "*.tga" "*.fbx" "*.wav" "*.mp4"
git add .gitattributes
git commit -m "Add LFS tracking"
```

> `.gitattributes` ファイルが作成されます。
> これをリポジトリに含めることで他の人も自動的にLFSを有効化できます。

☒ 5. SmartMerge（Unity YAML Merge）

① 目的

シーンやPrefabの競合を「自動で統合」してくれるUnity公式ツールです。
Gitに登録しておくと、複数人編集の事故が大幅減少します。

② 設定コマンド（最初の1回のみ）

Windows:

```
git config --global merge.unityyamlmerge.driver \
  "\"C:/Program Files/Unity/Hub/Editor/2022.3.XX/Editor/Data/Tools/UnityYAMLMerge.exe\" merge -p %O %A %B %A"
```

macOS:

```
git config --global merge.unityyamlmerge.driver \
  '"/Applications/Unity/Hub/Editor/2022.3.XX/Unity.app/Contents/Tools/UnityYAMLMerge" merge -p %O %A %B %A'
```

③ `.gitattributes` に対象拡張子を追加

```
*.unity      merge=unityyamlmerge text eol=lf
*.prefab     merge=unityyamlmerge text eol=lf
*.mat        merge=unityyamlmerge text eol=lf
*.anim       merge=unityyamlmerge text eol=lf
*.asset      merge=unityyamlmerge text eol=lf
```

☒ 6. 日常運用チェックリスト

チェック項目	内容
☐ Library フォルダが含まれていない	`.gitignore` で除外
☐ .meta ファイルを削除していない	参照切れを防ぐため
☐ Pull → 作業 → Commit → Push	作業前に最新状態を取得
☐ Additive Scene / Prefab 分業	同時編集の競合を避ける
☐ Unity バージョン統一	チーム内で固定する

☒ 推奨フォルダ構成（チーム開発用）

```
Assets/
  ├── _Scripts/
  ├── _Art/
  ├── _Audio/
  ├── _Prefabs/
  └── Scenes/
Packages/
ProjectSettings/
.gitignore
.gitattributes
README.md
```

☒ 参考リンク

- ☒ Unity公式: SmartMerge (YAMLMerge)
- ☒ Git LFS 公式サイト
- ☒ Microsoft Docs: Visual Studio Git 統合

典型的なトラブル回避 “7 か条”

1. **メタファイルを必ずコミット**（Visible Meta Filesが前提）。メタ欠落＝参照切れ地獄。
2. **1シーン＝1人原則**で作業（Additive SceneやPrefab分割で並行開発）。
3. **Packages/manifest.json と packages-lock.json をコミット**（依存固定）。
4. **外部DLL/SDKは UPM か `Assets/Plugins` に整理**（配布手順もREADMEに）。
5. **巨大アセットはLFS**、履歴を肥やさない。不要ファイルは**早めに`git rm`**。
6. **ブランチ運用**：`main` は常に動く／開発は `feature/xxx` ／PRでレビュー → マージ。
7. **GitHub 100MB制限**を意識（超過前にLFS。履歴に入った巨大ファイルは `git lfs migrate` で修正）。

初期セットアップ（手順書テンプレ）

1. プロジェクト新規作成 → 設定変更（Visible Meta / Force Text）
2. `.gitignore` と `.gitattributes` 配置、`git init` → 初回コミット
3. `git lfs install` → 大型拡張子を `git lfs track`
4. UnityYAMLMerge を `git config` に登録
5. GitHubにcreate repo → `git remote add origin ...` → `git push -u origin main`
6. チームへ**Unityバージョン・ブランチ運用・シーン分割規約**を共有

日々のワークフロー（短縮版）

- **作業前**：`git pull --rebase`
- **作業**：新規ブランチで、シーンはAdditive化・Prefab化して分業
- **コミット**：小さく早く。メタも含めて`git add -A`
- **PR**：スクショ/動画/GIF付きで意図を説明 → レビュー → マージ
- **衝突**：まずSmartMergeに任せる → ダメならUnityで開いて手作業修正

授業・組織運用の小ワザ

- **テンプレートリポ**（Starter Code）を作って、GitHub Classroomで配布
- **プリフック**（任意）：`Library/` をステージしようとしたら弾くpre-commit
- **命名規則**：`feature/ui-login`、`fix/physics-jitter` など
- **アセット置き場規約**：`Assets/_Art`, `Assets/_Audio`, `Assets/_Scripts` で迷子防止
- **READMEに最小セット**：Unityバージョン／導入手順／ブランチ戦略／LFS対象

Revision #4
★Created 6 October 2025 07:14:25 by youe2
Updated 2 June 2026 19:54:13 by youe2