Fachprojekt DET SS 2019
- Einführung in git -

Marco Pleines | Dortmund, 02.04.2019

Einführung in git

Marco Pleines | Dortmund, 02.04.2019

git

- Tool zur verteilten Versionierung von Quellcode
- Source Code verteilt auf Remotes versionieren
- Features/Bugfixes auf Branches entwickeln
- Ältere Commits wiederherstellen

- Ideal zum kollaborativen Arbeiten an einem Softwareprojekt

Clients und Server

- Git Terminal Client: z.B. Bash
- Git GUI Client: z.B. Sourcetree, GitKraken, Github Desktop
- Git Server: GitLab, GitHub

Beispiel: Benutzerkennung setzen

- *git config --global user.name “My Name”*
  - Name des Benutzers
- *git config --global user.email “mymail@mail.com”*
  - E-Mail des Benutzers

Beispiel: Commit erstellen

- *git init*
  - initialisieren eines Repositories
- *git status*
  - anzeigen von Dateien mit Änderungen
- *git add*
  - hinzufügen von Dateien zur Versionierung
- *git commit -m “my first commit”*
  - erstellen eines Commits der zu versionierenden Daten

Beispiel: Remote hinzufügen

- *git add remote origin https://github.com/user/repo.git*
  - fügt einen remote server namens origin hinzu
- *git add .*
  - zu versionierende Daten hinzufügen
- *git commit -m “my commit”*
  - Commit erstellen
- *git push origin master*
  - pushen des commits auf den master branch von origin

Beispiel: Änderungen vom Remote laden

- *git pull origin master*
  - Updates vom master branch von origin in die lokale Version überführen (fetch & merge)

Beispiel: Branches

- *git branch feature/newFeature*
  - erstellt Branch mit dem Namen feature/newFeature
- *git checkout feature/newFeature*
  - wechsel zum branch feature/newFeature
- *git branch*
  - listet alle vorhanden branches auf

Typischer Ablauf

- Projektdaten bearbeiten (git status)
- Daten zur Versionierung hinzufügen (git add)
- Commit mit den Änderungen erstellen (git commit -m “”)
- Ggf. Updates vom Remote mergen (git pull)
- Ggf. Merge-Konflikte beheben
- Remote auf den neuesten Stand bringen (git push)

Und wenn etwas schief geht …

… zurückspringen zu einem älteren Commit oder gemachte Änderungen zurückverfolgen.

Oder vorsorglich auf einem gesonderten Branch arbeiten.

Beispiel: .gitignore

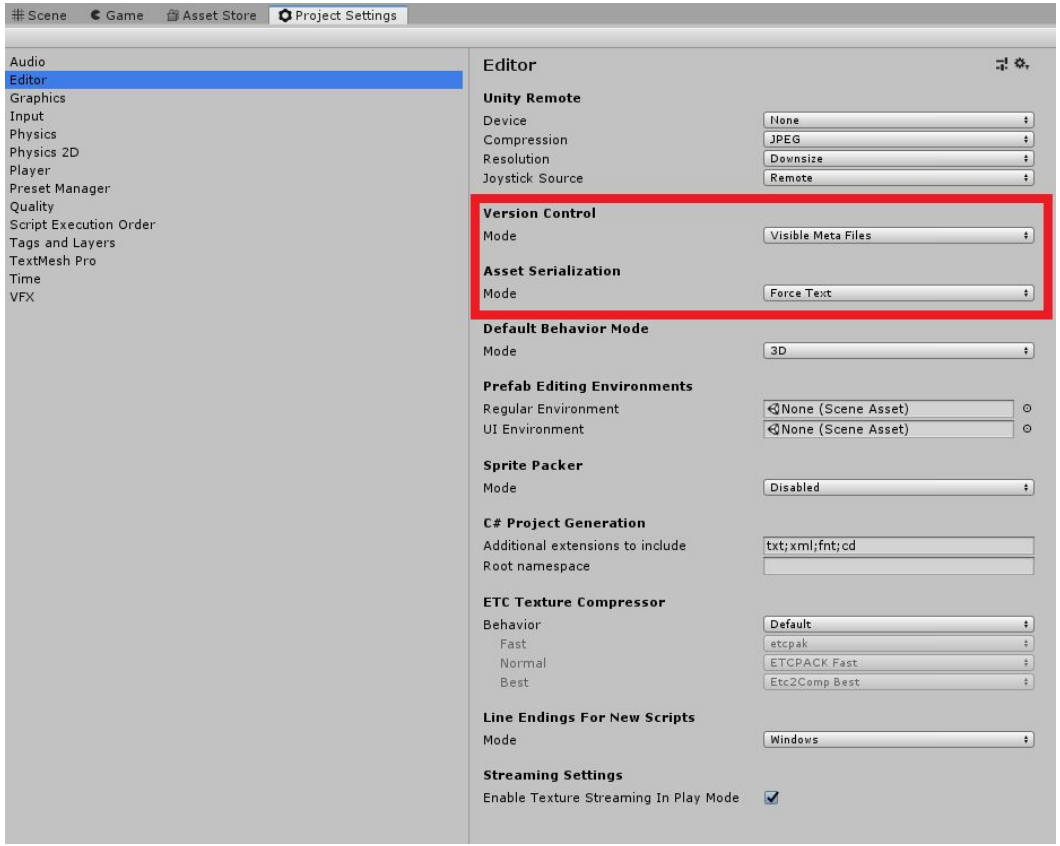
- Daten, die nicht versioniert werden sollen, werden in der .gitignore angegeben

```
/builds/
/bin/
/[Ll]ibrary
!readme.md
```

Unity & Git

- .gitignore für Unity
- Ganz wichtig!
  - Library Ordner ignorieren!

- Text Serialization
- Visible Meta Files



Git LFS

- Versionieren von großen Dateien (> 10mb)
- Anstatt der Dateien werden Zeiger zu der Datei genutzt
- Git LFS