Übungen für die Fortbildung

Inhalt

Für die Fortbildung sind Übungen mit GIT vorgesehen, die den Anwendungsszenarien für den Unterricht entsprechen. Dabei schlüpfen die Teilnehmenden sowohl in die Rolle der Lehrer als auch in die Rolle der Schüler. Die Fortbildenden sollten daher genau darauf achten, die jeweilige Rolle deutlich herauszuarbeiten.

Fortbildner: Das Arbeiten mit GIT ist sehr anspruchsvoll und für GIT-Anfänger oftmals sehr verwirrend. Beschränken Sie sich daher in der ersten Fortbildung auf die Übungen mit einem lokalen Repository (Kapitel Section 1, “Lokal Versionieren”), das Anlegen eines Online-Repositorys und des Sicherns des lokalen Repos auf dem Online-Speicher (Kapitel Section 2.1, “Vorlage übernehmen und lokal versionieren”), damit an verschiedenen Rechnern gearbeitet werden kann. Alle anderen Übungen sind optional und können ggf. in einer Anschlussveranstaltung durchgeführt werden.

1. Lokal Versionieren

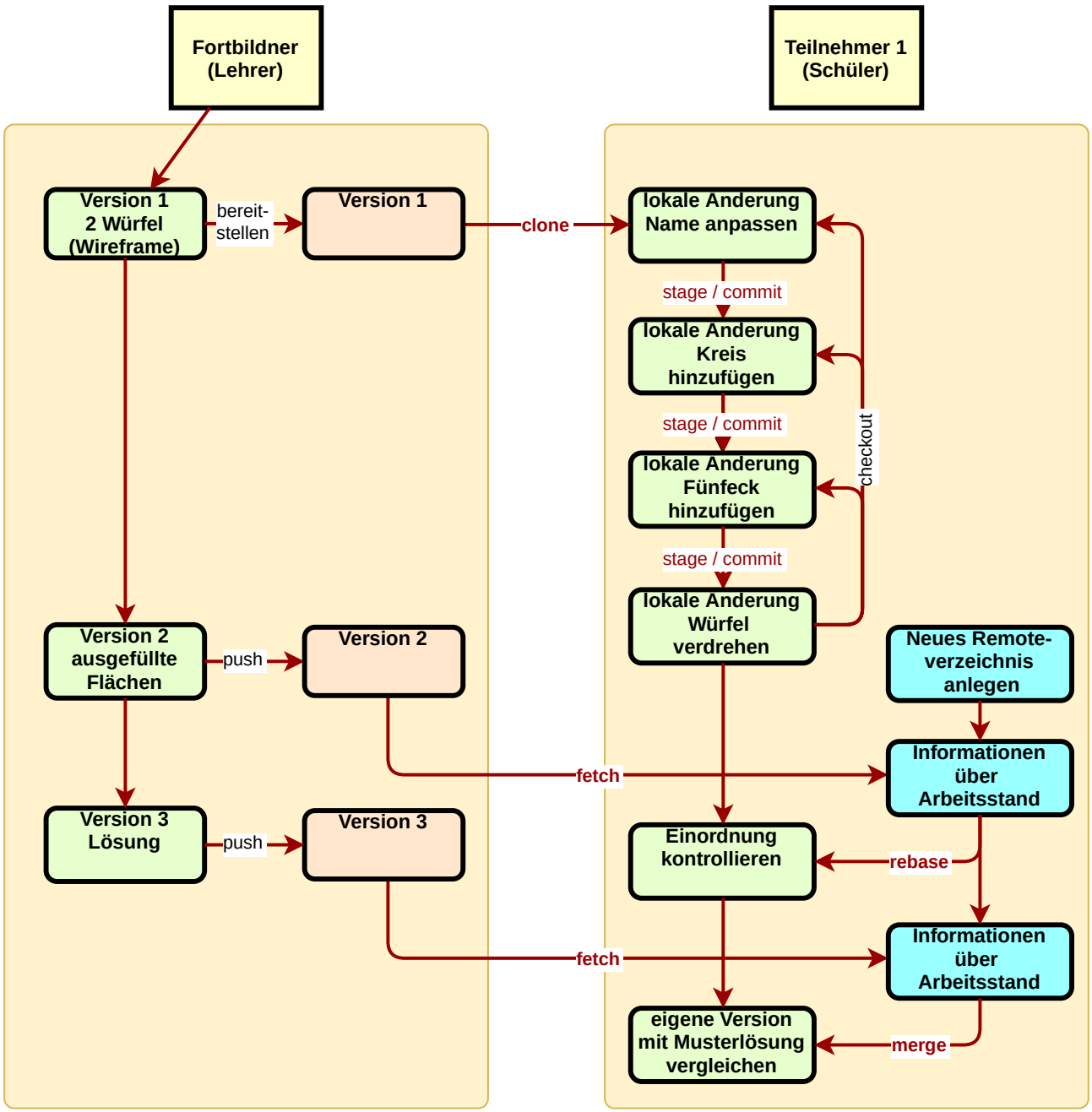
Fortbildner: Das lokale Versionieren wird in den folgenden Übungen noch intensiver geübt. Daher geht es hier nur darum eine erste Erfahrung zu sammeln und zu erkennen, wie eine Versionsverwaltung arbeitet.

Entscheiden Sie, ob der Arbeit mit einer GUI eine kurze Arbeitsphase mit der GIT-Konsole vorangestellt werden sollte.

Aufgaben:

1. Legen Sie mit Smartgit ein neues Repository an (Menü Repository→Add or create). Wählen Sie dort einen Ordner aus. Da er noch kein Repository enthält, müssen Sie das Repository zunächst initialisieren.
2. Erstellen Sie in diesem Ordner eine Text-Datei und schreiben Sie "Version 1" als Text hinein.
3. Sie sehen in Smartgit, dass eine Datei hinzugekommen ist. Erstellen Sie eine "Commit"-Nachricht, fügen die Datei dem Stage-Bereich hinzu und commiten diesen Zustand.
4. Verändern Sie die Textdatei (z. B. "Version 2" hineinschreiben) und legen Sie eine weitere Datei an. Commiten Sie diese Veränderungen auch.
5. Sie können nun zwischen den beiden Version hin- und herwechseln und sehen in Smartgit, welche Veränderungen gemacht wurden. Es biete sich an, die Schülerinnen und Schüler (mindestens) jeweils am Ende der Unterrichtsstunde einen Commit machen zu lassen, dann ist der jeweilige Arbeitsstand des Tages gesichert.

2. Mit Vorlagen in einem Online-Repository arbeiten



Ziel der zweiten Übung ist es, selbst ein Repository durch Clonen eines online bereitgestellten Repositorys anlegen zu können, mit diesem zu arbeiten und die Arbeitsschritte lokal zu versionieren.

In der Schule wird in der Regel, das vom Lehrer bereitgestellte Repository von den Schülern geclont. Sie befinden sich also im Moment in der Rolle der Schüler.

> **Vorbereitung Fortbildner:** Erstellen Sie ein neues Repository und kopieren Sie die Version 1 des Matter-Projekts dort hinein. Stellen Sie das Repository online bereit und teilen Sie den Teilnehmern den Link zum Repo mit.

2.1. Vorlage übernehmen und lokal versionieren

Aufgaben:

1. Erzeugen Sie ein neues lokales Repository durch Clonen des vom Fortbildner angegebenen Online-Repository.
2. Öffnen Sie die Datei Test.html mit einem Browser.
3. Öffnen Sie die Datei Test.html mit einem beliebigen Texteditor (z. B. notepad++). Fügen Sie Ihren Namen bei "Autor" hinzu. Speichern Sie die Datei, überzeugen Sie sich, dass die Änderungen im Browser zu sehen sind.
4. Sie sehen auch in Smartgit, dass die Datei verändert wurde und bekommen die Änderungen im Detail angezeigt. Tragen Sie eine Commit-Nachricht ein, stagen Sie die Datei und commiten Sie die Änderungen.
5. Fügen Sie einen Kreis (Bodies.circle verlangt die X- und die Y-Koordinate des Mittelpunkts und den Radius als Parameter) in der Welt hinzu.

   ```
   var circle = Bodies.circle(440, 150, 40);
   ```

   Beachten Sie, dass er auch beim Composite.add Befehl ergänzt werden muss. Stage/Commiten Sie die Änderungen.

6. Fügen Sie ein Fünfeck (Bodies.polygon verlangt die X- und die Y-Koordinate des Mittelpunkts, die Anzahl der Ecken und den Radius als Parameter).

   ```
   var fuenfeck = Bodies.polygon(380, 250, 5, 30);
   ```

   Stage/Commiten Sie die Änderungen.

7. Verdrehen Sie die Würfel um einen beliebigen Winkel:

   ```
   var boxA = Bodies.rectangle(400, 200, 80, 80, {angle: 0.5} );
   ```

   Stage/Commiten Sie die Änderungen.

8. Wechseln Sie in Smartgit zu den verschiedenen Entwicklungsstufen (checkout). Es reicht, wenn Sie ohne lokalen Branch (read only) arbeiten. Kontrollieren Sie im Browser, dass sich die Datei Test.html im jeweiligen Entwicklungszustand befindet.

2.2. Änderungen an der Vorlage (Rebase)

Es kommt immer wieder vor, dass ein Fehler in den bereitgestellten Dateien erst im Verlauf des Unterrichts bemerkt wird. Nun wäre es gut, die Originaldateien ändern zu können und

trotzdem alle von den Schülern gemachten Änderungen beibehalten zu können.

> **Vorbereitung Fortbildner:** Kopieren Sie Version 2 der Matter-Projekt-Vorlage in ihr Repo und pushen Sie die neue Version.

9. Sollte noch kein Remote-Verzeichnis in Smartgit existieren, legen Sie in Smartgit ein neues für das Online-Repo des Fortbildners an, das Sie vorhin geclont haben. Rufen Sie mit Fetch die Information über dieses Repo ab.
10. Führen Sie ein Rebase durch. Wählen Sie ihren lokalen Branch aus und rufen Sie das Kontextmenü auf. Achten Sie darauf, dass ihr lokaler Branch fett markiert ist. Wählen Sie dann "Rebase" aus und führen Sie das Rebase automatisch durch ("Rebase HEAD to").
    Kontrollieren Sie im Browser, dass die Objekte nur gefüllt sind. Untersuchen Sie, an welcher Stelle der Versionshistorie die Umstellung auf "wireframes: false" stattgefunden hat.

2.3. Musterlösung bereitstellen

Gerne möchte man als Lehrender am Ende oder als Zwischenstand eine Musterlösung bereitstellen. Die Schülerinnen und Schüler sollen ihre Lösungen mit der Musterlösung vergleichen und nur dort, wo sie nicht fertig wurden oder Fehler erkannt wurden, die Musterlösung übernehmen.

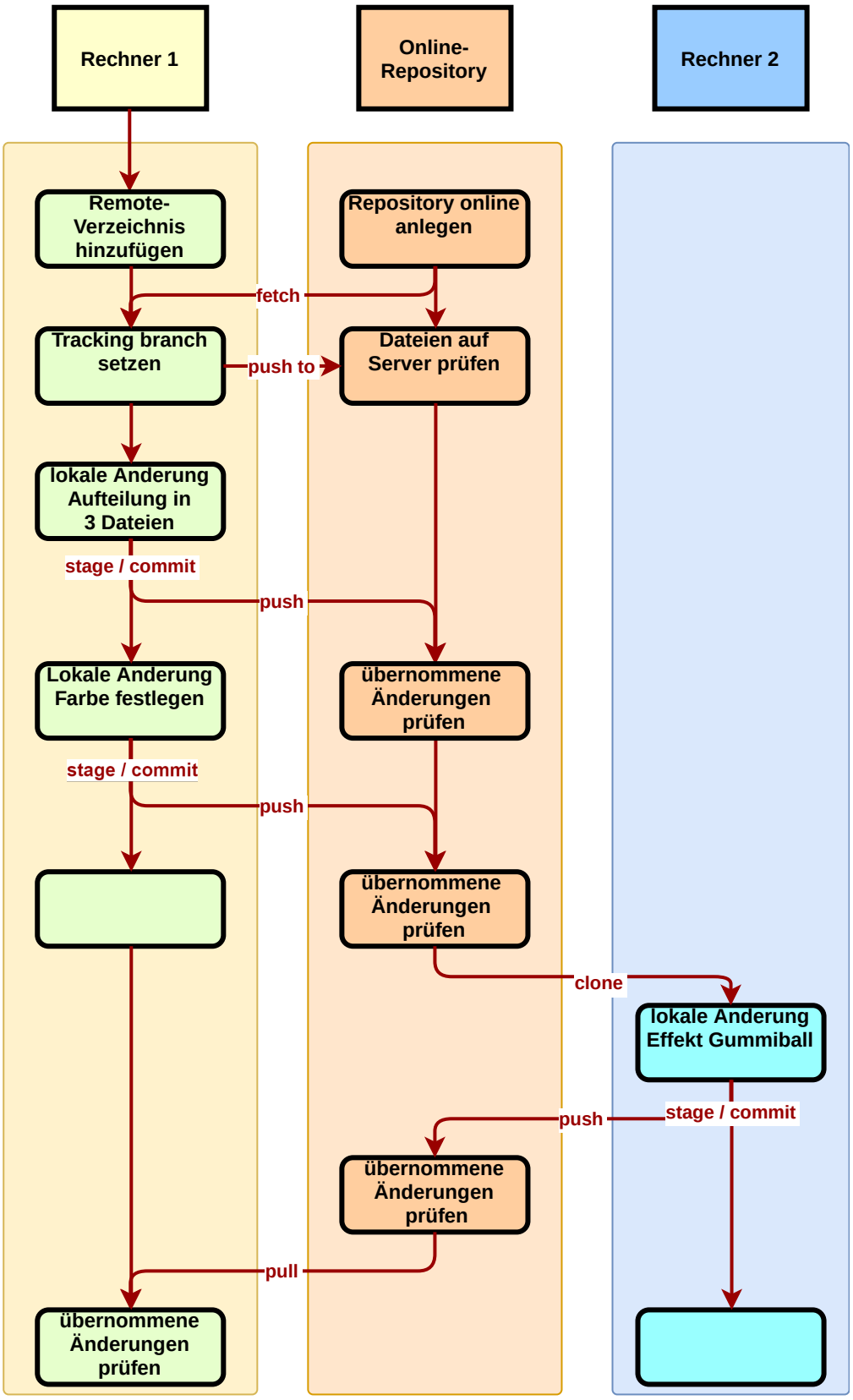
> **Vorbereitung Fortbildner:** Kopieren Sie Version "Würfel verdreht" der Matter-Projekt-Vorlage in ihr Repo und pushen Sie dieses.

11. Führen Sie erneut einen Fetch für das Online-Repo des Fortbildners durch.
12. Führen Sie nun über das Kontextmenü des lokalen Branches ein Merge durch. Wählen Sie dazu "Merge to Working Tree". Dies hat den Vorteil, dass die aktuellen Änderungen jederzeit verworfen werden können, wenn der Merge Probleme bereitet.
    Vergleichen Sie dabei die Musterlösung der Fortbildungsmaterialien mit ihrer Lösung. Sie sehen, dass die Datei Test.html sowohl von Ihnen wie auch von der Musterlösung verändert wurde (The file is in Conflicted (both modified) state). Rufen Sie den Conflict Solver auf. Links sehen Sie ihre Lösung (ours), rechts die heruntergeladene Musterlösung (theirs). Sie können nun über die ">>" bzw. "<<" Buttons entscheiden, welche Variante sie übernehmen wollen. Übernehmen Sie nur dann die Musterlösung, wenn ihre Lösung Fehler enthält oder unvollständig ist. Speichern Sie die aktualisierte Version, schließen Sie den Conflict Solver und markieren sie den Konflikt als gelöst.
    Zum Abschluss müssen Sie noch einen Commit durchführen, um den Merge-Prozess abzuschließen.

3. Eigenes Repository online ablegen

Ein lokales Image ermöglicht die Versionsverwaltung auf einem Rechner. Sowohl Schüler als

auch Lehrkräfte arbeiten aber sowohl in der Schule als auch zu Hause. Daher ist ein Datenaustausch zwischen verschiedenen Rechnern notwendig. Dieser erfolgt über die Speicherung auf einem GIT-Server. Das Land Baden-Württemberg stellt dafür den GIT-Camp Server zur Verfügung. In dieser Übung lernen Sie den Datenaustausch mit dem GIT-Camp Server kennen.



3.1. Anlegen eines Repositories auf dem GIT-Camp Server

1. Loggen Sie sich mit Ihren GIT-Camp Login-Daten auf dem Server ein.
2. Legen Sie dort ein neues Repository an:
   - Wählen Sie einen Namen.
   - Stellen Sie die Sichtbarkeit auf "privat".
   - Geben Sie eine kurze Beschreibung ein : z. B. "Matter-Projekt der GIT-Fortbildung".
   - Die restlichen Felder können Sie leer lassen. Der Haken bei "Repository initialisieren" darf **nicht** gesetzt sein.
3. Kopieren Sie die URL des Repositorys und legen Sie in Smartgit in Ihrem Repository ein neues Remote-Verzeichnis (=Online-Repository) an (Menü Remote→Add). Fügen Sie die kopiere URL ein und wählen als Namen "GIT-Camp".
4. Führen sie zunächst ein Fetch auf diesem Remote-Verzeichnis durch, damit Smartgit die Informationen über das Repositiory abfragt. Führen Sie danach ein "Push to..." mit ihrem lokalen Repository durch und wählen das Remote-Verzeichnis als Ziel. Prüfen Sie auf "Git-Camp", dass die Dateien übertragen wurden.
5. Damit ihr lokales und das Remote-Verzeichnis verknüpft sind, können Sie außerdem "Set tracked branch" auf ihr Remote-Verzeichnis setzen. Sie müssen dann in Zukunft nicht immer angeben, mit welchem Remote-Verzeichnis ein pull oder ein push durchgeführt werden soll. Dazu muss allerdings vorher einmal ein Push auf dieses Remote-Verzeichnis durchgeführt werden.

Sie führen nun noch eine weitere Änderungen an den Dateien durch und legen diese auf dem Server ab.

6. Teilen Sie die Datei Test.html in drei Teile auf:
   - Schneiden Sie aus dem JavaScript-Teil alles aus, was mit den Objekten in der Welt zu tun hat (Erstellen der Objekte und Einfügen in die Welt) aus und fügen Sie es in eine neue Datei "objekte.js" ein. Ein umgebender <script>-Tag ist nicht notwendig.
   - Schneiden Sie aus dem JavaScript-Teil alles aus, was mit der Welt zu hat (also den gesamten Rest des Scripts) und fügen Sie es in eine neue Datei "welt.js" ein.
   - Ändern Sie "Test.html" so ab, dass die beiden Scripte geladen werden:

     ```
     <script src="welt.js"></script>
     <script src="objekte.js"></script>
     ```

     Der umgebende Rest an HTML-Code bleibt erhalten.
7. Stage/Commiten Sie die Änderungen lokal. Pushen Sie die Änderungen auf den Git-Camp-Server.

8. Wählen Sie die Farben der Objekte selbst aus, z. B.:

   ```
   boxA.render.fillStyle = "#FF66AC"; // z.B. für Pink
   ```

   Stage/Commiten Sie die Änderungen lokal. Pushen Sie die Änderungen auf den Git-Camp-Server.

9. Überzeugen Sie sich, dass die verschiedenen Versionen nun auf dem Git-Camp-Server verfügbar sind und die gesamte Änderungshistorie nachvollziehbar ist (auch die Änderungen, die erfolgt sind, bevor das Repository auf den Git-Camp-Server hochgeladen wurde).

3.2. Arbeiten an verschiedenen Orten

In dieser Übung sollen Sie lernen, wie Sie an verschiedenen Orten mit ihrer jeweils aktuellen Version arbeiten können. Wechseln Sie daher mit ihrem Nachbarn den Rechner.

1. Clonen Sie ihr eigenes Repository auf dem Rechner des Nachbarn. Wählen Sie dazu in Smartgit Menü Repository→Clone und geben die URL ihres Repositories auf dem Git-Camp-Server an. Wählen Sie als lokales Verzeichnis ein neues, leeres Verzeichnis.
2. Überzeugen Sie sich, dass ihre aktuelle Version nun lokal auf dem neuen Rechner verfügbar ist. Untersuchen Sie, ob auch alle vorherigen Versionen vorhanden sind.
3. Die Objekte soll nun wie ein Gummiball herumspringen. Dazu muss der Boden gummiartig werden:

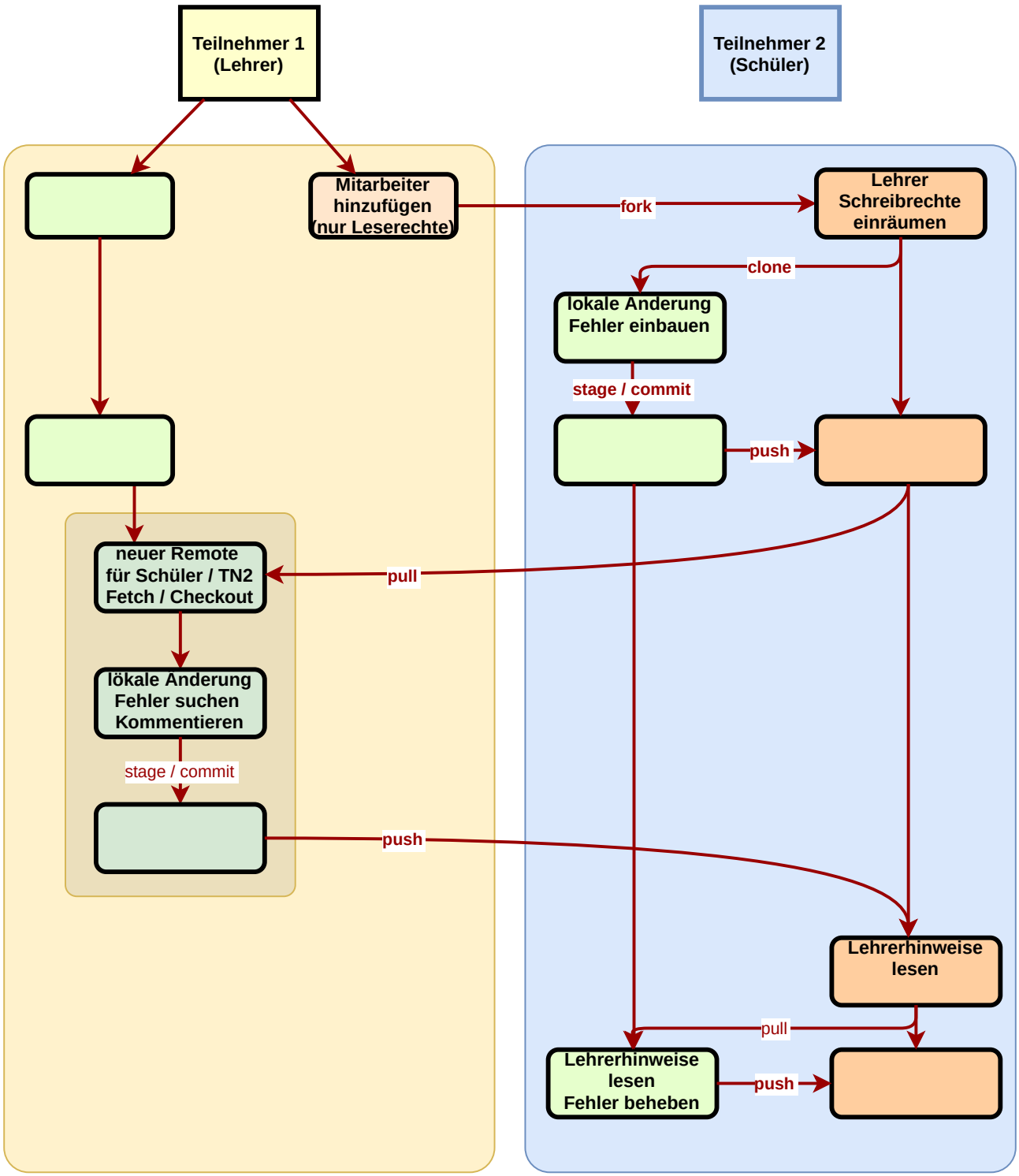
   ```
   ground.restitution = 1.3;
   ```

   Stagen/Commiten Sie die Änderung lokal und pushen Sie die Änderung auf den Git-Camp-Server. Überzeugen Sie sich davon, dass die Änderungen auf dem Git-Server übernommen wurden.

4. Wechseln Sie zu ihrem eigenen Rechner zurück und pullen Sie aktuelle Version vom Git-Camp-Server. Überzeugen Sie sich, dass die Objekte jetzt auch auf diesem Rechner herumspringen.

4. Zusammenarbeit Lehrer-Schüler

In dieser Übung wird die Situation simuliert, dass die Schüler mit einem online-Repository arbeiten, das auf einer Vorlage des Lehrers beruht, und der Lehrer den aktuellen Arbeitsstand der Schüler sichtet und Fehler behebt oder zumindest kommentiert.



Dazu ist es notwendig, dass der Schüler lesenden Zugriff auf das Repository des Lehrers und der Lehrer lesenden und schreibenden Zugriff auf das Repository des Schülers besitzt. Nur so kann er die Korrekturen an den Schüler zurückgeben.

In dieser Übung agieren Sie sowohl als Lehrer als auch als Schüler: Als Lehrer stellen Sie ein Repository bereit und gewähren den Schülern lesenden Zugriff. "Schüler" ist in diesem Fall Ihr linker Nachbar. Dieser legt einen Fork[1] ihres Repositorys an und baut einen Fehler in das Programm ein. Sie übernehmen diesen Part für Ihren rechten Nachbarn. Danach holen Sie sich das Repositorys des "Schülers" (ihres linken Nachbarn) und suchen dessen Fehler. Die korrigierte und kommentierte Version geben Sie an den "Schüler" zurück.

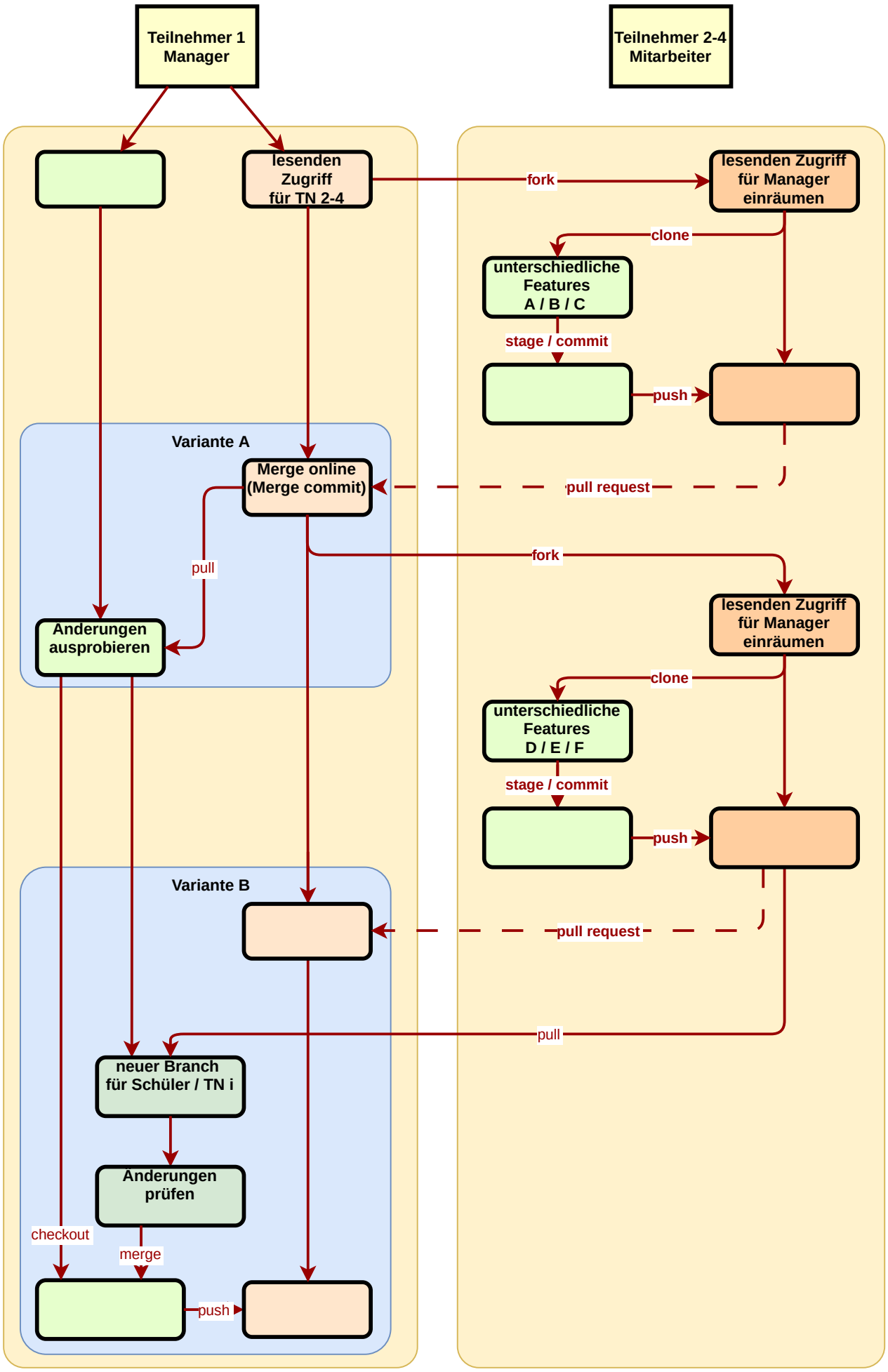
1. Lehrerrolle: Räumen Sie ihrem linken Nachbarn (dem "Schüler") im Git-Camp Leserechte auf Ihr Online-Repository ein. Gehen Sie dazu in der Onlineumgebung von Git-Camp beim Repo auf Einstellungen→Mitarbeiter und fügen den Namen ihres linken Nachbarn hinzu. Gewähren Sie nur Leserechte! In der Schule müssten Sie das für die ganze Klasse machen. (TODO: Geht das auch für Gruppen?)
2. Schülerrolle: Da Ihr "Lehrer" (rechter Nachbar) sein Repository für Sie freigegeben hat, sehen Sie es in Git-Camp in der Liste der verfügbaren Repositories. Wählen Sie es aus und legen Sie einen Fork an. Benennen Sie den Fork so, dass Sie Ihren Namen an den Repository-Namen anhängen.
3. Schülerrolle: Räumen Sie Ihrem "Lehrer" (rechter Nachbar) Lese- und Schreibrechte auf Ihren Fork ein. Im Schulumfeld ist dieser Schritt nicht notwendig, da Sie als Besitzer einer Gruppe in der Organisation automatisch Lese- und Schreibrechte auf die Repos der Gruppenmitglieder besitzen.
4. Schülerrolle: Legen Sie eine lokale Kopie Ihres Forks an, indem Sie das Repository mit SmartGit clonen.
5. Schülerrolle: Bauen Sie an beliebiger Stelle einen Fehler ein. Stage/commiten Sie die geänderte Version und verwenden Sie als Commit-Message "Ich weiß nicht weiter… Es geht nichts mehr." (Sie dürfen das "nichts" auch näher beschreiben, dass sollte man von Schülern verlangen).
6. Lehrerrolle: Legen Sie (für jeden Schüler) im Smartgit ein neues Remote-Verzeichnis an (Menü Remote → Add). Geben Sie dort die URL des Forks Ihres "Schülers" (rechter Nachbar) an. Sie können in Git-Camp alle Forks ihrer Schüler anzeigen lassen, wenn Sie auf die Zahl neben "Fork" klicken. Wählen Sie einen Fork aus, sehen Sie die URL des Forks im Browser.
   Als Name des Remote-Verzeichnis wählen Sie nicht "origin", sondern den Namen Ihres Schülers.
7. Lehrerrolle: Holen Sie sich die Informationen über das neue Remote-Verzeichnis, indem Sie ein "Fetch" auf dem Remote-Verzeichnis ausführen (Rechtsklick). Sie sollten dann die Branches sehen, die das Projekt enthält (in der Regel nur einen master-Branch). Wählen Sie diesen aus, indem Sie ein "Check out" durchführen. Sie bestätigen dann die Frage, ob ein lokaler Branch angelegt werden soll. Auf diese Weise bekommen Sie für jeden Schüler einen Branch, der dem Namen des Schülers entspricht. Sie können dann sehr schnell zwischen den verschiedenen Schülerversionen hin- und herwechseln.
8. Lehrerrolle: Suchen (und korrigieren) Sie den Fehler, den Ihr Schüler eingebaut hat. Schreiben Sie eine Rückmeldung / Erklärung in die Commit-Message, welche Arbeiten der Schüler noch selbst durchführen muss, um seinen Fehler zu beheben. Stage/Commiten Sie die Änderungen lokal und pushen Sie die Änderung Sie Änderungen in den Fork des Schülers. Normalerweise müssen Sie dazu einfach "Push" wählen, da bei dieser Variante der Branch so angelegt wurde, dass der "Tracking branch" automatisch das Remote-Verzeichnis des entsprechenden Schülers ist. Ansonsten müssen Sie den Tracking branch setzen oder "Push to" benutzen.
9. Schülerrolle: Schauen Sie auf Git-Camp nach, ob Ihr "Lehrer" (rechter Nachbar) ihren Fehler gefunden hat und Ihnen Tipps gegeben hat, wie Sie ihn korrigieren können. Pullen Sie die Änderungen des "Lehrers", so dass ihr lokales Repository wieder auf dem aktuellen Stand

ist. Beheben Sie mit den Tipps des Lehrers ggf. noch vorhandene Fehler.
Achtung: Es ist essentiell, dass die SuS zunächst den Pull durchführen und den Fehler sich nicht nur online anschauen. Wird kein Pull durchgeführt und der Fehler lokal behoben, entsteht beim nächsten Push ein Konflikt, da eine neuere Version auf dem Server vorhanden ist. Diese muss erst gepullt und mit der lokalen Version gemerged werden.

5. Kooperatives Arbeiten

Die komplexeste Form der Nutzung von GIT ist das kooperative Arbeiten an einem Projekt. Dafür ist GIT konzipiert worden. Hier arbeiten mehrere Personen gleichzeitig an einem Projekt. Eine genaue Absprache, wer welche Teile des Projektes bearbeitet, ist unerlässlich. Trotzdem lässt es sich nicht vermeiden, dass einige Dateien von mehreren Personen bearbeitet werden. Es werden also beim Zusammenführen unweigerlich Konflikte auftreten.

Um das Chaos in einem überschaubaren Rahmen zu halten, sollten nicht alle Gruppenmitglieder volle Schreibrechte auf das Repository haben. Es ist einfacher, wenn zunächst jeder im eigenen Fork arbeitet und ein Gruppenmanager die Ergebnisse kontrolliert zusammenführt.


Teilnehmer 1
Manager
Teilnehmer 2-4
Mitarbeiter
lesenden
Zugriff
für TN 2-4
fork
lesenden Zugriff
für Manager
einräumen
clone
unterschiedliche
Features
A / B / C
stage / commit
push
Variante A
Merge online
(Merge commit)
pull request
pull
fork
Anderungen
ausprobieren
lesenden Zugriff
für Manager
einräumen
clone
unterschiedliche
Features
D / E / F
stage / commit
push
Variante B
pull request
pull
neuer Branch
für Schüler / TN i
Anderungen
prüfen
checkout
merge
push

1. Finden Sie sich in Gruppen von 3-4 Personen zusammen. Jeder von Ihnen ist im schulischen Kontext ein Schüler oder eine Schülerin. Bestimmen Sie einen Gruppen-Manager. Dieser ist verantwortlich dafür, die Arbeiten der einzelnen Gruppenmitglieder zusammenzuführen und online zu stellen.
2. Gruppenmanager: Als Gruppenmanager räumen Sie den Gruppenmitgliedern lesenden Zugriff auf ihr Repository ein.
3. Gruppenmitglieder: Forken Sie das bereitgestellte Repository und räumen Sie ihrem Gruppenmanager lesenden Zugriff auf ihren Fork ein.
4. Gruppenmitglieder (+ Manager): Bearbeiten Sie arbeitsteilig eine der folgenden Aufgaben. Sie sind aufsteigend nach Schwierigkeitsgrad geordnet. Nutzen Sie das Internet oder eine KI, um den Code zu erstellen.
   Tipp: Drücken Sie in Chrome Strg+Shift+I um die Entwickler-Tools angezeigt zu bekommen. Nur dort stehen Fehlermeldungen der Webseite, wenn Programmierfehler auftreten:
   - Lassen Sie die Geschwindigkeiten (velocity) der Objekte in der Welt durch kleine Pfeile anzeigen. Dies ist eine Funktionalität, die im Renderer aktiviert werden kann.
   - Fügen Sie rechts und links Wände hinzu, die sich natürlich nicht bewegen. Die Wände müssen nicht bis ganz oben gehen, so dass die Objekte ab einen gewissen Füllgrad über die Wände hinausfallen.
   - Bauen Sie Balken in der Welt ein, die sich nicht bewegen und auf denen die Objekte herunterrutschen. Die Reibung sollte also nicht zu groß sein.
   - Fügen Sie 1000 kleine Würfelchen in die Welt ein. Diese sollen an zufälligen Positionen in der oberen Hälfte des Bildschirm erzeugt werden. Schöner sieht es aus, wenn auch die Verdrehung zufällig ist.
     Wenn man die Reibung (friction) der Würfel verringert, dann rutschen sie allmählich auseinandern.
   - Fügen Sie der Webseite Schieberegler hinzu, die die Schwerkraft der Simulation in X- und in Y-Richtung verändern. Ordentlich beschriftet sollen sie natürlich auch sein.
   - Fügen Sie der HTML-Seite einen "Wasserhahn"-Button hinzu, der fortlaufend kleine blaue Kreise (Wassertropfen) in der Welt erzeugt, solange die Maus auf dem Button gedrückt gehalten wird.
   - Fügen Sie der HTML-Seite einen "Kanone"-Button hinzu, der größere Kreise von einer bestimmten Position aus verschießt. Die Kreise sollten zunächst schräg nach oben fliegen. Gegebenenfalls können Sie die Stärke des Schußes zufällig wählen oder irgendwie steuren. Wenn das Gewicht der Kreise groß ist, können sie andere Objekte wegschießen.
5. Gruppenmitglieder: Nachdem Sie eine Aufgabe erfüllt haben, stagen/commiten Sie die Änderungen. Pushen Sie die Änderungen auch in ihr Online-Repository. Gehen Sie dann zur Online-Umgebung von Git-Camp wählen Ihr Repository aus. Wählen Sie dann den Reiter "Pull-Request" und erzeugen Sie einen neuen Pull-Request. Das Ziel des Requests muss das Repository des Gruppenmanagers sein. Von Ihrem eigenen Repository soll gepullt werden. Sie sehen, welche Änderungen im Vergleich zum Manager Sie vorgenommen haben.
   Der Manager erhält dadurch eine Nachricht (ggf. sogar eine E-Mail), dass Sie die

Einarbeitungen der Änderungen wünschen. Sie sollten daher dem Pull-Request einen sinnvollen Titel und eine Beschreibung Ihrer Arbeit hinzufügen. Andernfalls wird der Manager den Request vermutlich gar nicht erst in Betracht ziehen.

6. Gruppenmanager: Sie sehen in Git-Camp, dass ein Pull-Request eingetroffen ist.
   - Variante A: Sie können dem Pull-Request direkt online stattgeben und die Repos zusammenführen. Dies ist für sehr einfache Änderungen (z. B. Tippfehler) sinnvoll. Holen Sie sich im Anschluß mit Pull die neue Version in ihr lokales Repo.
   - Variante B: Größere Änderungen sollten vorher begutachtet werden. Dann sollten Sie zunächst das Repo des Gruppenmitgliedes herunterladen und lokal zusammenführen.
     Fügen Sie wie in der Übung "Zusammenarbeit Schüler-Lerher" ein neues Remote-Verzeichnis für das Repo des Gruppenmitgliedes hinzu. Führen Sie einen Fetch durch und checken Sie dann den master-Branch des neues Remote-Verzeichnisses out. Stimmen Sie dem Anlegen eines lokalen Branches zu. Wählen Sie diesen lokalen Branch aus (check out) und testen Sie die Neuerungen.
     Wenn Sie mit diesen Neuerungen einverstanden sind, dann wählen Sie ihren eigenen lokalen Master-Branch aus (check out). Dieser muss unbedingt fett geschrieben und mit einem Pfeil markiert sein! Wählen Sie dann im Kontext-Menü des Branches mit den Neuerungen den Menüpunkt "Merge" (Merge to Working Tree) aus. Dort müssen Sie nun gegebenenfalls Konflikte mit dem Conflict Solver lösen.
     Schließen Sie den Merge-Prozess ab, indem Sie eine sinnvolle Commit-Message schreiben und die Änderungen sichern. Pushen Sie die Änderungen in Ihr Online-Repo.
7. Gruppenmitglieder: Wenn Sie eine Aufgabe erfüllt und den Pull-Request erstellt haben, dann können Sie eine weiter Aufgabe in Angriff nehmen. Erstellen Sie dazu einen neuen Fork, da Sie ja jetzt eine neue Aufgabe bearbeiten. Hat der Gruppenmanager schon eine neue Version hochgeladen, haben Sie auch alle Neuerungen im neuen Fork dabei (auch Ihre eigenen). Wählen Sie also eine weitere Aufgabe aus der obigen Liste aus oder denken Sie sich etwas eigenes aus.

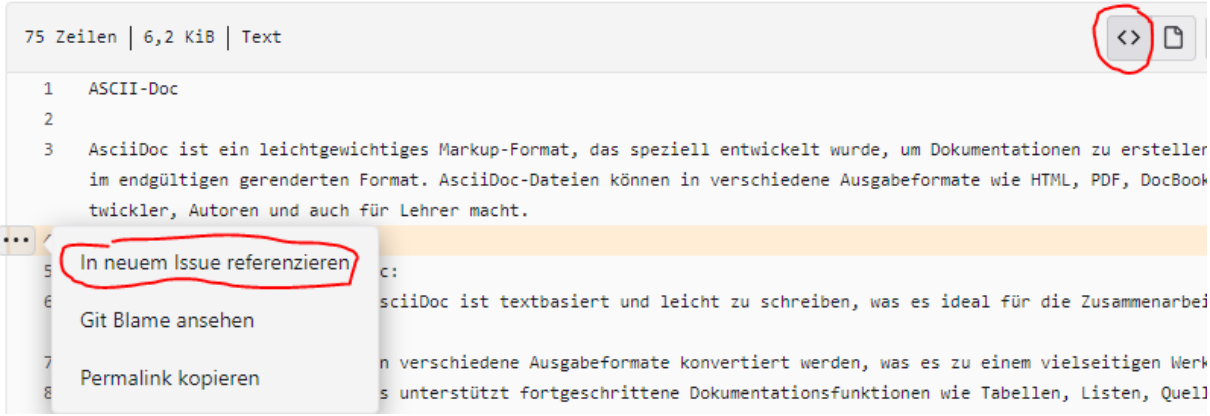
6. Mit ZPG-Materialien arbeiten (Verändern / Fehler melden usw.)

Auch die Zusammenarbeit mit uns Fortbildnern soll in Zukunft über GIT vereinfacht werden: Sie können entweder Fehler und Anregungen an uns melden oder selbst Fehler beheben, Verbesserungen implementieren oder Texte überarbeiten.

1. **Fehler melden:** Wählen Sie in Git-Camp das Repo mit unseren Fortbildungsmaterialien aus. Vielleicht haben Sie ja schon einen Tippfehler gefunden oder eine andere Ungenauigkeit festgestellt. Dann melden Sie dies nun. Wählen Sie den Reiter "Issues". Legen Sie einen neuen Issue an. Wenn Sie keinen Fehler gefunden haben, hinterlassen Sie einfach einen (netten) Kommentar für uns.

   Dieser Issue kann an eine bestimmte Zeile im Code gebunden werden. Das erleichtert den Fortbildnerinnen die Arbeit. Dazu lassen Sie sich im Gitcamp den Quellcode einer Datei

anzeigen, so dass links Zeilennummern zu sehen sind (Quellcode muss explizit rechts oben gewählt werden!). Dann kann in der fehlerhaften Zeile ganz links geklickt und ein Issue zu dieser Zeile erstellt werden.



2. **Fehler selbst beheben:** Sie sollen das bereitgestellte Dokument Asciidoc.adoc überarbeiten. Da ist mit der Formatierung noch einiges schief gelaufen. Arbeiten Sie bitte genauso wie in Übung 5 und erstellen Sie einen Fork der Materialien zu dieser Fortbildung, arbeiten Sie die Änderungen an Asciidoc.adoc ein und stellen Sie danach einen Pull-Request. Der oder diejenigen Veranwortlichen für die Materialien werden sich den Request anschauen und eine Rückmeldung geben. Wird er angenommen, haben Sie vielleicht für viele Lehrer zu einer Verbesserung des Unterrichts beigetragen. Sollte er abgelehnt werden, können Sie trotzdem ihre Änderungen selbst verwenden.

   Sollte später einmal ein Request abgelehnt werden, müssen Sie trotzdem Änderungen der Originalmaterialien mit Ihren (abgelehnten) Änderungen zusammenführen. Sonst profitieren Sie entweder nicht mehr von den Verbesserungen am Originalmaterial oder verlieren Ihre eigenen Änderungen.

   Führen Sie dazu folgende Schritte durch:

   - Legen Sie lokal ein neues Remote-Verzeichnis mit dem Repo der Fortbildungsmaterialien an. Führen Sie dort einen Fetch durch und checken Sie dann den master-Branch des neues Remote-Verzeichnisses out. (Vgl. Übung 4)
   - Wählen Sie ihren eigenen lokalen Master-Branch aus (check out). Dieser muss unbedingt fett geschrieben und mit einem Pfeil markiert sein! Wählen Sie dann im Kontext-Menü des Branches der Originalmaterialien den Menüpunkt "Merge" (Merge to Working Tree) aus. Dort müssen Sie nun gegebenenfalls Konflikte mit dem Conflict Solver lösen.
     Schließen Sie den Merge-Prozess ab, indem Sie eine sinnvolle Commit-Message schreiben und die Änderungen sichern.

[1] Diese Anwendungsszenario lässt sich auch mit einem Clone umsetzen. Allerdings ist ein Clone normalerweise dafür vorgesehen, wenn man zusammen an einem einzigen Projekt arbeiten möchte. Jeder Schüler/jede Schülerin arbeitet aber an einem eigenen Projekt. Daher sollte hier ein Fork gewählt werden.