Aufbau interaktiver 3D-Engines

Universität Osnabrück
Fachbereich Mathematik / Informatik

2. Übung

Prof. Dr. rer. nat. Oliver Vornberger
Nico Marniok, B. Sc.
Erik Wittkorn, B. Sc.

16.04.2013

Übersicht

1. Git Reloaded
2. Besprechung des 1. Übungsblattes
3. Engine-Repository
4. Engine/TestBed Installation
5. TileGame Struktur (Preview)

git reloaded

- Tipps und Tricks
  - Alias anlegen
    - 'git config --global alias.co checkout'
    - 'git config --global alias.br branch'
    - 'git config --global alias.ci commit'
    - 'git config --global alias.st status'
    - *Beliebiger Alias möglich*
- Git Genesis
  - '*git init*' - initialisiert ein neues repository im akt. Ordner

git reloaded

- Änderungen am repository nachverfolgen
  - Gilt nur für getrackte Dateien
  - '*git diff*' - zeigt den Unterschied zwischen "staging area" und working directory (noch nicht gestaged)
  - '*git diff --staged*' - zeigt den Unterschied zwischen "staging area" und letztem commit
- Weitere Änderungen vornehmen
  - '*git mv Test.txt KeinTest.txt*' - Datei umbenennen
  - '*git rm Test.txt*' - Datei löschen und dies im commit merken
  - '*git rm --cached Test.txt*' - Datei aus der Versionsverfolgung nehmen

git reloaded

- Änderungen rückgängig machen
  - '*git commit --amend*' - kann die neue "staging area" dem letzten commit hinzufügen
  - '*git reset HEAD Test.txt*' - Änderung aus der "staging area" nehmen
  - '*git checkout -- Test.txt*' - Änderungen im "working directory" zur "staging area" rückgängig machen
  - Bei einem Merge alles mit der eigenen Version überschreiben: '*git reset HEAD .*' && '*git checkout -- .*' && '*git commit -am "merge: my version is just better"* '

git reloaded

- Git History durchforschen
  - '*git log -x*' - die letzten x commits anzeigen
    - '*--pretty=oneline*' - Übersichtlicher in einer Zeile
    - '*--stat*' - Statistiken des commits
    - '*--graph*' - grafische Darstellung der Branches
    - '*--since=2.weeks*' - Alle commits der letzten zwei Wochen
    - '*--grep=fehler*' - Suche in den commit-messages
  - '*gitk*' - grafische Oberfläche

```
3625b92e6c6737251dec3c6a270bdb32e4050259 neuste Version
1f2aea041cfb78e3757cd41ce79a3077d4a691b4 doublemerged
dbdd3df7ecedd6048f7b0b6c767a69c98772fdde merged
26813706e6c2654699356a3174a9c1dc506b9f0f merged
323e7eba67ee1c4b901fee200eccbe48fadff322 andere Zeile
3df5daf54528044eb7748dd9cc042da7b207c9df Mittlere Zeile
7bc12b2041069c6ce108fc7d907ea69df2aeb212 Zweite Zeile
c05a6bc59d0ae365f541881ec4bb77868567a0cb Erste Zeile
127e58a9226eed39efd90ebe195ea65a546aee01 wieder geloescht
```

reloaded

DEMO

git reloaded

- Basic branching and merging
  - Ein branch ist eine Referenz auf einen commit
  - Man legt einen branch an, um den Produktionscode nicht zu stören
    - Neues Feature
    - Experimente
  - Merge (zusammenführen) erst nach Test
  - '*git branch -a*' zeigt alle existierende branches an
  - '*git push origin master*' -> '*git push origin branch*'

git reloaded

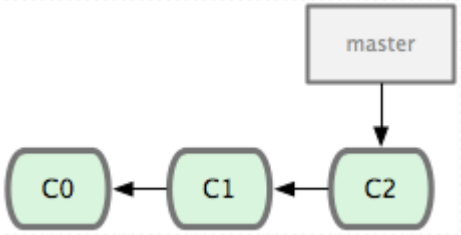
1. Man möchte an einem Projekt arbeiten
2. Man erstellt einen neuen branch für das aktuelle Arbeitspaket
3. Man arbeitet in diesem branch

Es muss ein Hotfix beim Produktions-Code durchgeführt werden

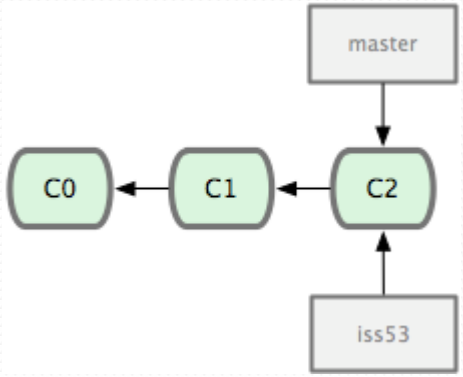
1. Wechseln zum master-branch
2. Hotfix-branch erstellen und den Hotfix implementieren
3. Nach einem Test den Hotfix zum master-branch mergen
4. Wieder zum eigentlichen Arbeitspaket wechseln

git reloaded

- Basic Branching and Merging

1. 

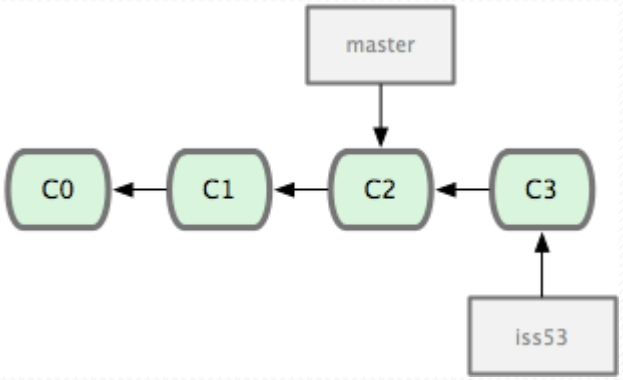
2. *'git checkout -b iss53'*



git reloaded

3. *'vim Test.txt'*

   'git commit -am "cool new feature" '



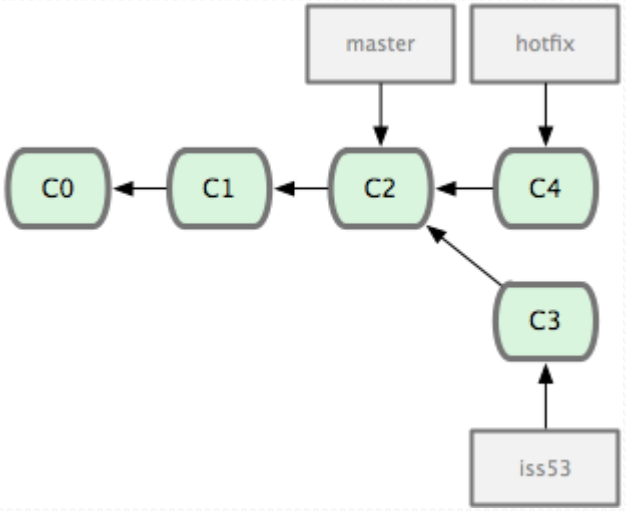

NASA
905
United States
HOUSTON
WE HAVE A PROBLEM

git reloaded

1. *'git checkout master'*
2. *'git checkout -b hotfix'*

   'vim Test.txt'

   'git commit -am "removed some bugs" '



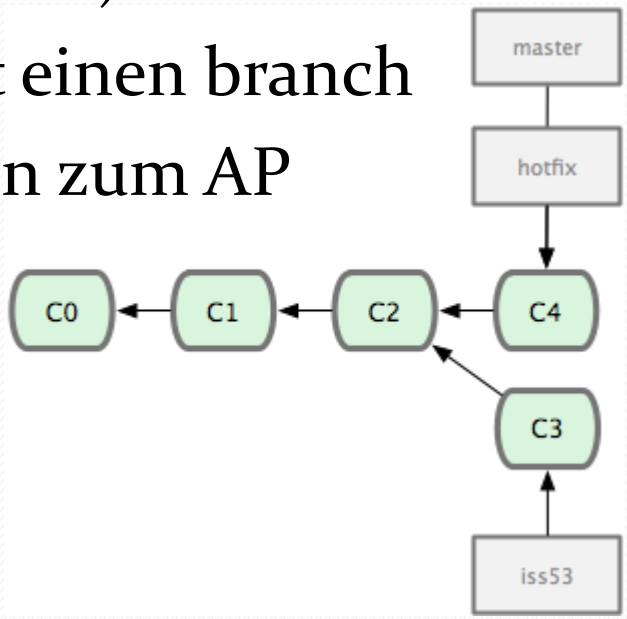
git reloaded

3. '*git checkout master*'

'*git merge hotfix*' - fügt Änderungen eines anderen branchs hinzu (eventuell Konflikte)

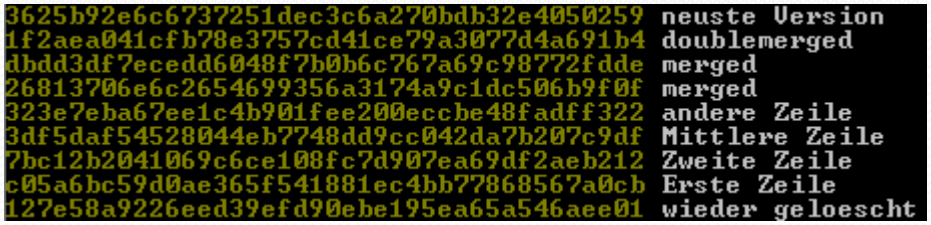
'*git branch -d hotfix*' - löscht einen branch

4. '*git checkout iss53*' - wechseln zum AP



git reloaded

- *Zu älterer Version zurückgehen*
  - *'git log --pretty=oneline'*



1. Alte Version in neuem Branch angucken
   - *'git checkout -b alte-version 26813'*
2. Aktuellen branch zurücksetzen (obacht)
   - *'git reset --hard 26813'*
   - 'git reflog' - Hashcode dieses Wechsels wiederbekommen

reloaded

DEMO

Besprechung von Übungsblatt 1

1. Anmerkungen
2. Kurzpräsentation nächstes Mal

Engine Repository

- Verfügbar unter git@vm205.rz.uos.de:ai3de-engine.git
- Regelmäßig neue Version auf unserer Page http://www-lehre.inf.uos.de/~ai3de/uebung.html Stud.IP (git ist aktueller)
- WIP = Work In Progress ☺
- Verbesserungsvorschläge bitte per Mail(ingliste) ;)

Engine/Testbed Installation

1. Neues Projekt "Engine" mit bestehendem Code von "*ai3de-engine*" erstellen
2. Libraries aus "*ai3de-uebung/libs/engine*" und "*lwjgl_utils.jar*" und "*lwjgl.jar*" (mit eingestellten natives)
3. Neues Projekt "TestBed" mit bestehendem Code von "*ai3de-uebung/engineprojects/TestBed*" erstellen
4. Engine-Projekt zum BuildPath hinzufügen
5. TestBed.java ausführen
6. Info/Error lesen:

```
[INFO] Config.java, 86: Config file config.ini does not
exist, so it will be created.
[ERROR] DirectoryResourceFile.java, 40: xxx/TestBedData/
does not exist
```

Engine/Testbed Installation

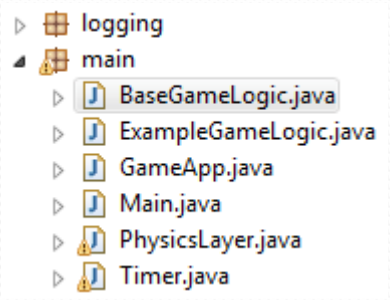
7. "TestBed" aktualisieren und in der config.ini die Zeile "*sResourcePath=xxx/TestBedData/*" mit dem richtigen Pfad zu "*ai3de-uebung/TestBedData*"
8. Yodas Zunge finden! ☺ (von http://thefree3dmodels.com/)

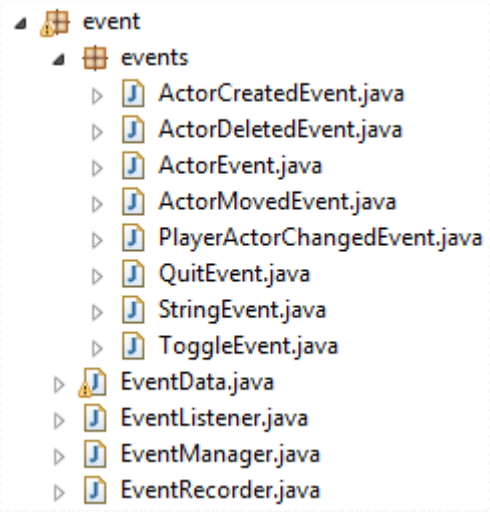
Engine/Testbed Installation

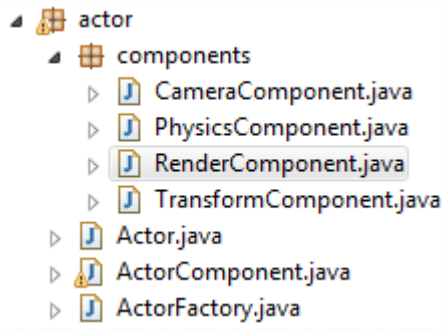
DEMO

Engine Repository

- Interessante Packages

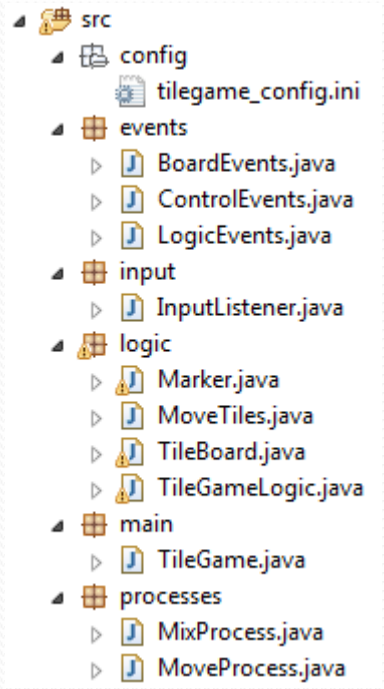






- ai3de-engine/TestBedData/Levels/testbed.xml definiert die Szene
- In der config die Tastenbelegung ändern

TileGame Struktur (Preview)




```java
public static class ActorMoveEvent extends ActorEvent {

    public static final int ID = 0xe0562b29;
    private Vector3f position = new Vector3f();

    public ActorMoveEvent(int actorId) {
        super(actorId);
    }

    public void setPosition(float x, float y, float z) {
        this.position.set(x, y, z);
    }

    public Vector3f getPosition() {
        return this.position;
    }

    @Override
    public int getId() {
        return ID;
    }
}
```

TileGame Struktur (Preview)

- TileGameLogic.java implements EventListener

```
public void trigger(EventData data) {
    //TODO What if Actor is null ?
    Actor actor = null;
    switch (data.getId()) {
        case LogicEvents.ActorMoveEvent.ID:
            ActorMoveEvent actorMove = (ActorMoveEvent)data;
            actor = this.getActor(actorMove.getActorId());
            if (actor != null) {
                TransformComponent transform =
(TransformComponent)actor.getComponent(TransformComponent.ID);
                transform.getPose().setPosition(actorMove.getPosition());
            }
            break;
    }
}
```

Nächste Woche:

- Präsentation des aktuellen Übungsblattes
- Eigene Events definieren
- Vorstellung des nächsten Übungsblattes

Vielen Dank für die Aufmerksamkeit ☺