Aufbau interaktiver 3D-Engines

Universität Osnabrück
Fachbereich Mathematik / Informatik

1. Übung

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

Organisatorisches

- Übung: Dienstags, 12:00 – 14:00, regelmäßige Teilnahme erwünscht!
  - Gehalten von Erik Wittkorn
  - Erste Übung am 09.04.2013!
- Testate alle 2 Wochen in 2er Teams
  - Ausgabe Übungsaufgaben: Alle 2 Wochen dienstags, ab 09.04.2013 (1. Übung)
  - Abgabe Lösungen: Alle 2 Wochen freitags, ab 19.04.2013, 23:59
  - Vorstellung einer Lösung alle 2 Wochen dienstags in der Übung ab 23.04.2013
- Größeres Projekt am Ende der Vorlesungszeit über 2 Wochen in 2er Teams
  - Zwischenpräsentation nach einer Woche vor Erik und Nico
  - Endpräsentation nach zwei Wochen vor Herrn Vornberger
  - Tipp: Bereitet die Endpräsentation *sehr gut* vor und haltet die zur Verfügung stehende Zeit *unbedingt* ein!

Organisatorisches

- Terminplanung Abschlusspräsentation
  - 11. Vorlesung/Übung am 24.06.2013/25.06.2013
  - 12. Vorlesung/Übung am 01.07.2013/02.07.2013
  - 13. Vorlesung/Übung am 08.07.2013/09.07.2013
    - Vermutlich Klausurenwoche
  - In den Semesterferien

Übersicht

1. Einführung in Eclipse/Netbeans
2. Versionsverwaltung mit git
3. Logging
4. XML-Parsing mit Java

Einführung in  /  NetBeans

- Download und Installation von “Eclipse for Java Developers” unter http://www.eclipse.org/downloads/ oder “NetBeans IDE Java SE” unter https://netbeans.org/downloads/
- Aufgaben-Code kopieren und ein neues Projekt mit bestehendem Code erstellen
- Einbinden von Bibliotheken
  - Lwjgl http://www.lwjgl.org/
  - Lwjg_util
- Pfad für LWJGL-Natives setzen
- Hotkeys nutzen und eigene definieren

Was ist Versionsverwaltung?

- Verwaltung verschiedener zeitlicher Versionen eines (Software)-Projekts
- Reduziert Ausmaß von Fehlern
- Wird für (alle) OpenSource-Projekte im Internet verwendet
- Probleme bei Team-Arbeit
  - Unübersichtlicher Datenverkehr per Mail
  - Dropbox?!!!!
    - Kein gleichzeitiges Arbeiten an denselben Dateien möglich
    - Keine Wiederherstellung von alten Zuständen möglich
    - Keine Archivierung alter Zustände
    - Keine Protokollierung der Änderungen
    - Keine Sicherheit der Daten

Versionsverwaltung mit 

- Verwaltung eines Arbeitsverzeichnisses (*repository*) lokal oder von einem Server
- Jeder Nutzer hat die ganze History eines Projekts
- Verschiedene Versionszweige eines Projekts sind möglich
- Kostenlos und für alle Betriebssysteme erhältlich

Versionsverwaltung mit git

- Bereiche

Lokal

Server

working directory

staging area

repository

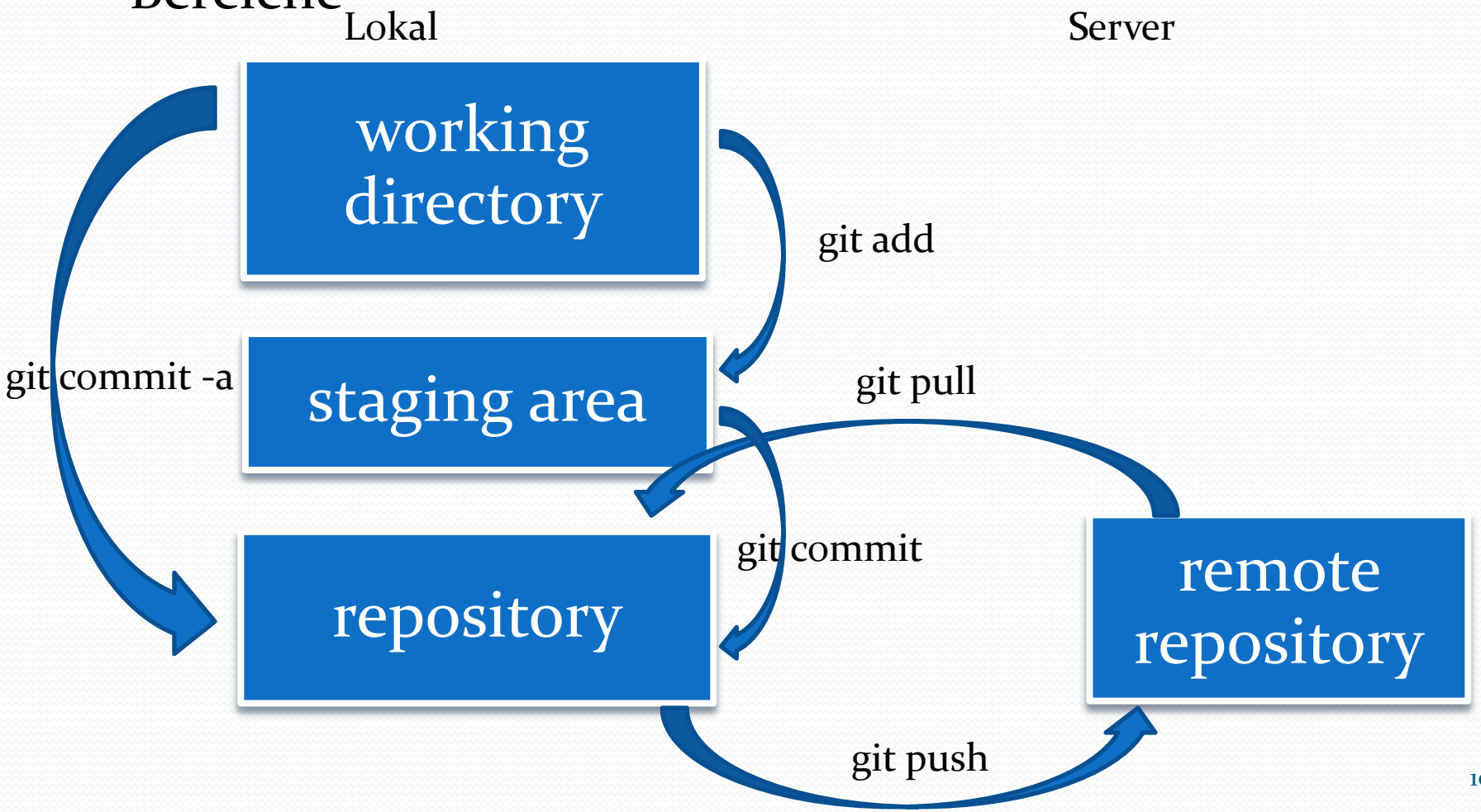
remote repository

Versionsverwaltung mit git

- Wichtige Befehle
  - *'git clone repository.git'* – Eine lokale Arbeitskopie eines Repositories erstellen
  - *'git status'* – Den aktuellen Status des Repositories abfragen
  - Dateien ändern, hinzufügen
  - *'git add 42.txt'* – Eine neue Datei hinzufügen
  - *'git commit –am "Habe die Frage gefunden"'* – Änderung abschließen
  - *'git pull origin master'* – Änderungen vom Remote-Repository ins lokale Repository und working directory integrieren
  - *'git push origin master'* – Änderungen dem Remote-Repository mitteilen

Versionsverwaltung mit git

- Bereiche



Versionsverwaltung mit 

DEMO

Versionsverwaltung mit git

- Weitere Besonderheiten
  - Merge-Konflikte
  - Authentifizierung per ssh-Keys
  - .gitignore-Datei mit Namen der zu ignorierenden Daten
  - 'git add' bei neuen Dateien nicht vergessen
  - Abgabe der Übungsaufgaben per git!!

- Wichtige Links
  - git http://git-scm.com/
  - github https://github.com/
  - git extensions http://code.google.com/p/gitextensions/

Logging

- Singleton
- Features:
  - Schreibt Logs in Datei und gibt sie auf der Konsole aus
  - Automatische Angabe der Datei und Zeile einer Ausgabe
  - Verschiedene Prioritäten
- Logger.INSTANCE.info("Eine gute Info!");

- Hinweise
  - Bei Eclipse ein Editor-Template anlegen (z.B. linfo)
  - In Netbeans ein Code-Template anlegen

Logging

Der Output:

```
[INFO] KeineAhnung.java, 42: Eine nette Info
[WARNING] KeineAhnung.java, 42: Eine kleine Warnung
[ERROR] KeineAhnung.java, 42: Ein gefaehrlicher Fehler
```

Soll nacher so aussehen:

```
[INFO] Main.java, 23: Eine nette Info
[WARNING] Main.java, 25: Eine kleine Warnung
[ERROR] Main.java, 27: Ein gefaehrlicher Fehler
```

XML-Aufbau

- <Start-tag>
- </End-tag>
- <Empty-element-tag />
- <!-- Kommentar -->
- Attribute mit name="value"
- <Element> <Kindelement1 /> <Kindelement2> </Kindelement2> </Element>

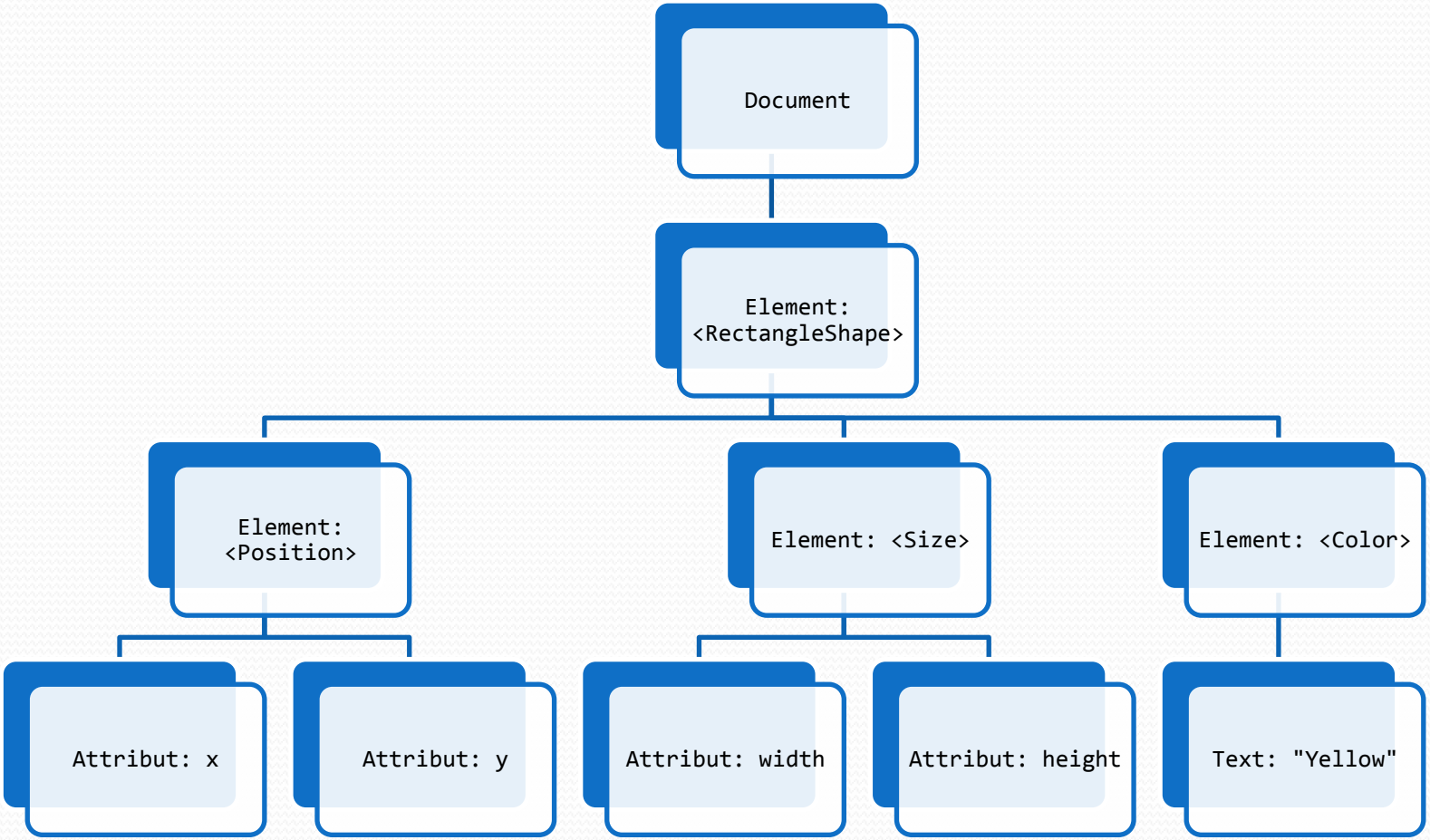
```
<RectangleShape>
  <Position x=„4.0“ y=„3.0“ />
  <Size width=„2.0“ height=„7.0“ />
  <Color>Yellow</Color>
  <!-- Wozu ist dieses Rechteck da? -->
</RectangleShape>
```

DOM-Parsing

- Document Object Model - baumbasiert
- Einheitliche Schnittstelle für HTML- und XML-Dokumente
- Baumstruktur mit verschiedenen Knoten
  - Dokumentknoten
  - Elementknoten
  - Attributknoten
  - Textknoten
- Vorteile
  - Navigation zwischen einzelnen Knoten des Dokuments möglich
  - Man kann Knoten Erzeugen, Verändern und Löschen
- Nachteile
  - Das ganze Dokument wird eingelesen
  - Langsamer und benötigt mehr Speicher

XML als Baum

```
<RectangleShape>
  <Position x=„4.0“ y=„3.0“ />
  <Size width=„2.0“ height=„7.0“ />
  <Color>Yellow</Color>
</RectangleShape>
```



DOM-Parsing Code

```
<RectangleShape>
  <Position x=„4.0“ y=„3.0“ />
  <Size width=„2.0“ height=„7.0“ />
  <Color>Yellow</Color>
</RectangleShape>
```

```
Document xmlDoc = dBuilder.parse(xmlFile);
Node rectangleShapeNode = xmlDoc.getFirstChild();

while (rectangleShapeNode != null) {
    Node rectangleShapeChildNode = rectangleShapeNode.getFirstChild();

    while (rectangleShapeChildNode != null) {
        switch (rectangleShapeChildNode.getNodeName()) {
            case "Position": // Die Attribute des <Position>-tags werden eingelesen
                float x = Float.parseFloat(rectangleShapeChildNode.getAttributes().getNamedItem("x").getNodeValue());
                float y = Float.parseFloat(rectangleShapeChildNode.getAttributes().getNamedItem("y").getNodeValue());
                break;

            case "Size":
                float width = …, height = …
                break;

            case "Color":
                String color = rectangleShapeChildNode.getTextContent(); // Die Farbe befindet sich im TextContent der Node
                break;
        }
        rectangleShapeChildNode = rectangleShapeChildNode.getNextSibling(); // Die "Geschwister"-Node wird eingelesen
    }
    rectangleShapeNode = rectangleShapeNode.getNextSibling(); // Die "Geschwister"-Node wird eingelesen
}
```

SAX-Parsing

- Simple API for XML - eventbasiert
- Einheitliche Schnittstelle für XML-Dokumente
- XML wird sequentiell gelesen
- Man definiert *callback functions* für verschiedene Ereignisse
  - startElement("Position", [{"x":"4"}, {"y":"3"}])
  - characters("yellow")
  - endElement("RectangleShape")
- Werden im *Document-Handler* definiert
- Vorteile
  - Schneller und benötigt weniger Speicher
  - Kann auch trotz Fehler weiterarbeiten
- Nachteil
  - Kein direkter Zugriff auf Elemente oder Änderung am Baum

SAX-Parsing Code

```
<RectangleShape>
  <Position x=„4.0“ y=„3.0“ />
  <Size width=„2.0“ height=„7.0“ />
  <Color>Yellow</Color>
</RectangleShape>
```

```
private static String currentString = null; // Platzhalter-Referenz für Element-Content
public void startElement(String uri, String localName, String qName, Attributes atts) throws SAXException {
    switch (localName) {
        case "Position":  // Die Position der Form wird aus den Attributen des <Position>-Tags ausgelesen
            float x = Float.parseFloat(atts.getValue("x"));
            float y = Float.parseFloat(atts.getValue("y"));
            break;
        case "Size": // Die Groesse der Form wird aus den Attributen des <Size>-Tags ausgelesen
            float width, height = …
            break;
        case "Color": // beim Start-Tag <Color> wird eine default-Farbe gesetzt
            currentString = "defaultColor";
            break;
    }
}
public void characters(char[] ch, int start, int length) throws SAXException {
    currentString = new String(ch, start, length);
}
public void endElement(String uri, String localName, String qName) throws SAXException {
    switch (localName) {
        case "Color": // Der Name der Farbe muesste sich in der aktuellen String-Variable befinden und wird gesetzt
            rectangleShape.setColor(currentString);
            break;
    }
}
```

XML-Parsing

DEMO

Übungsblatt 1

1. Aufgabe: git
2. Aufgabe: OpenGL-Test-Applikation
3. Aufgabe: Logging
4. Aufgabe: XML-Parsing

Nächste Woche:

- Besprechen des Lösungsfortschritts
- Simples Branching in git

Vielen Dank für die Aufmerksamkeit ☺