Dokumentation Zork

Von Loris Hütter und Aaron Holenstein

27.1.2022

Loris Thierry Hütter & Aaron Holenstein

BBW 5IA20b

Inhalt

[1. Projektplanung 2](#_Toc94171301)

[1.1. Testingkonzept mit JUnit 2](#_Toc94171302)

[1.2. Git Vorgehensmodell 2](#_Toc94171303)

[1.3. Clean/-Code Struktur 2](#_Toc94171304)

[2. Abschluss 2](#_Toc94171305)

[2.1. Reflexion Loris Hütter 2](#_Toc94171306)

[2.2. Reflexion Aaron Holenstein 3](#_Toc94171307)

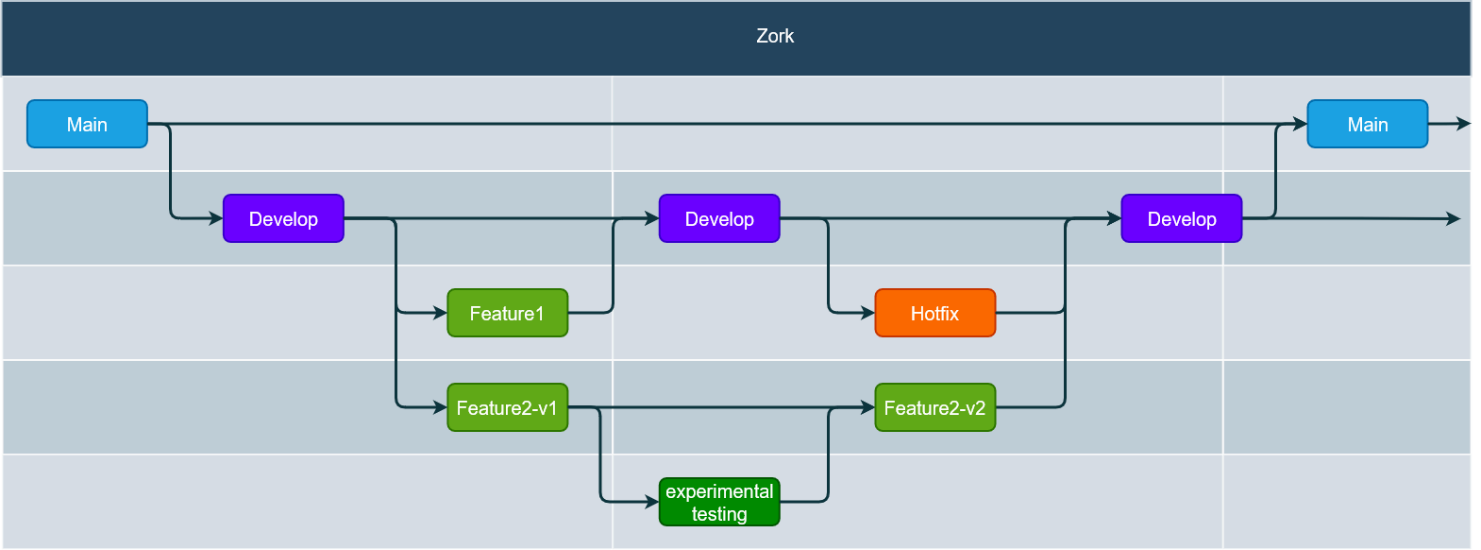
# Projektplanung/ Vorgehen

# Testingkonzept mit JUnit

Die Tests wurden so geschrieben, dass eigentlich alle Methoden, welche einen Rückgabewert haben, getestet werden. Hierbei wurden sowohl blackbox, als auch whitebox tests verwendet.

# Git Vorgehensmodell

Bei unserer Git Struktur ist der Develop Branch der aktivste Branch. Wenn ein Feature entwickelt wird, wird ein neuer Branch erstellt, der auf den Develop Branch basiert. Ein Feature Branch kann auch einen Experimentellen Sub Branch haben, falls Beim schon bearbeiteten Feature drastische unsichere Änderungen vorgenommen werden sollen und bei negativem Ergebnis, kann auf den normalen Feature Branch zurückgegriffen werden. Es können an zwei Features parallel gearbeitet werden, die beide in den Develop merged werden. Dabei werden alle möglichen Merge-Errors bereinigt. Wenn der Feature zu Develop merge abgeschlossen wird, wird der Feature Branch gelöscht. Wenn beim Develop ein Problem auftaucht, kann ein Hotfix Branch erstellt werden, in dem diese Probleme gelöst werden und zurück mit dem Develop merged werden. Wenn der Develop getestet und stabil funktioniert wird er mit dem Main merged.



# Clean/-Code Struktur

Wir haben uns an die standard Clean-Code Regeln gehalten, ausserdem alle Variabeln und Methoden im Camel-Case.

# Vorgehen

# Diagram Description automatically generated2.1 Map

Diese Karte war ursprünglich geplant, doch mussten wir dann eine simplere Version umsetzen

Wir haben uns die Arbeit immer in der Schule vorausgeplant und einzelne Tasks zugeteilt.Dann haben wir bei jedem merge erneut den Fortschritt und das weitere Vorgehen besprochen. Ausserdem haben wir per Teams call und screen share einander geholfen, wenn jemand nicht weiterkam.

# Abschluss

# Reflexion Loris Hütter

Der Auftrag war viel herausfordernder als ich ursprünglich dachte. Da das Projekt und sehr viel Spass bereitete und wir sehr motiviert waren, hatten wir auch sehr hohe Ziele uns gesetzt. Ich wollte, dass das Projekt gute Qualität ist und auch Spass mach für ein Text-Adventure. Wir konnten trotzdem alle Core-Features einbauen. Am meisten Probleme bereitete mir Git und die Merge-Conflicts. Ich verbrach knapp 3-4 Stunden im Total daran, diese conflicts zu fixen.

# Reflexion Aaron Holenstein

Der Auftrag hat sich als mehr Arbeit herausgestellt, als ich gedacht hatte. Dies lag vor allem an meinen hohen Erwartungen an meinen Code und auch daran, dass ich die bereitgestellten Klassen angepasst, bzw. verbessert habe. Meine Ideen in einem Game zu verwirklichen machte mir Spass und motivierte mich auch, kreative Lösungen zu finden.

Probleme bereitete mir vor allem das Verwalten der Räume. Schlussendlich habe ich mich dazu entschieden, die Räume in einem 2-Dimensionalen Array zu speichern und mittels einem Koordinatensystem auf sie zuzugreifen.

Der Schliessmechanismus von den Türen war eine weitere Hürde für mich, was ich dann aber schlussendlich lösen konnte, indem ich das Schliessen der Lock-Klasse zugewiesen habe, um diese dann als Objekt der Tür zu übergeben.