Diary

King of Jawa

Isabel Geissmann, Jannik Jaberg

Stand 6. Mai 2018

Eintrag vom 6. Mai 2018

- Source code is sufficiently documented
  Wir haben den Code mit dem Google Checkstyle überprüft und die fehlenden Javadocs hinzugefügt.
- Committed
  Wir werden heute Abend bzw. Morgen alles was committed wurde überprüfen damit wir nichts vergessen.
- GUI
  Unser Game kann in dem GUI gespielt werden.
- Referee/Victory
  Die Regeln des Games werden überprüft und es ist möglich das Spiel zu Gewinnen bzw. zu verlieren.
- Shall we play a game
  Diesen Punkt haben wir gestern zu viert überprüft und haben erfolgreich mehrere Runden gespielt.
- Unit-Test
  Nachdem wir alle Klassen gemerged haben, haben wir weniger Code Coverage als erwartet. Wir wollten mindestens eine «wichtige» Klasse komplett mit Unit Test abgedeckt. Wir haben uns dabei für die Chain Klasse entschieden.

Eintrag vo 5. Mai 2018

Heute sind wir soweit, wir können das erste richtige Spiel spielen, indem es möglich sein sollte zu gewinnen.
Zu Beginn hatten wir einige Probleme, nachdem alle in der Lobby bereit waren und das Spiel gestartet wurde stürzte ein Client ab. Dieses Problem konnten wir schnell beheben, nicht alle von uns haben daran gedacht gradle clean und build vor dem Testspiel auszuführen. Nach den anfänglichen Schwierigkeiten konnten wir unser erstes Spiel spielen und Pascal hat gewonnen.
Wir haben noch einige andere Bugs gefunden, im Lobby Fenster wird der State nur angezeigt, wenn dieser gewechselt wird währenddem der andere Client auch in der Lobby ist ansonsten wird das Feld einfach leer angezeigt. Die Gebäude können noch nicht gelevelt werden.
Uns ist auch aufgefallen, dass wir die Highscore noch einbinden müssen, damit der Gewinner auch in dem Highscore angezeigt wird.
Pascal wird die Highscore noch einbinden und die Rangliste im Chat anzeigen, nachdem ein Spiel beendet wurde. An den Bonuspunkten haben wir weniger als geplant gearbeitet, der Fullscreen im Spiel funktioniert jedoch bereits mit der Tastenkombination Alt und Enter.

Optimierungen

Wir müssen einiges an unserem Code optimieren und ändern um damit weiterarbeiten zu können.

- Lobby:
  Die Lobby zeigte sich nach dem Meilenstein 3 noch ein bisschen buggy. Weiter müssen wir sie optimieren für den späteren Spectator-Mode.
  - Liste aller Spieler in der Lobby
  - Spielerverwaltung
  - Chaträume
  - Joinbarkeit
  - Map bereit/nicht Bereit
  - Zuschauerliste
  - Synchronisation:
    * Spieler
    * Zuschauer
    * Map
- User:
  Für die User haben wir uns einen neuen Aufbau überlegt. Um später den Reconnect bei allfälligen Verbindungsverlusten oder RageQuits einfach zu handeln. Dafür haben werden wir die User ID mit einer UUID ersetzen, um dem Client die Möglichkeit zu bieten bei einem disconnect dem Spiel wieder zu joinen.
  - Session
  - UUID
  - Current Ping
- Session:
  Witer für den Reconnect implementieren wir ein Session-System.
  - Socket
  - IP
  - Current Ping
- Entities:
  - Buildins (Diese drei Building Kategorien können beliebig erweitert werden.)
    * Inhabitant Building
    * Ressource Building
    * Amusement Building

Eintrag vom 24. April 2018

Für diesen Meilenstein haben wir uns die Arbeiten aufgeteilt und arbeiten nicht wie sonst immer alle gemeinsam. Pascal und Nikolai werden den Code nochmals durchgehen und einige Optimierungen vornehmen, das Balancing einbinden und den Metropolenstatus implementieren damit es möglich ist das Spiel zu gewinnen.

- Referee
- Victory
- QA
  Damit wir unser Play-Testing durchführen können benötigen wir einen Bug-Tracker der auch von externen Personen genutzt werden kann. Aus diesem Grund haben wir uns gegen den Bug-Tracker von GitLab entschieden. Wir nutzen den Mantis Bug-Tracker, jeder der unser Spiel testet kann sich ein Konto erstellen und die gefundenen Bugs eintragen.

Jannik und Isabel kümmern sich um die Unit-Tests und werden alle nötigen Dokumente für den nächsten Meilenstein zusammenstellen.

- Unit-Tests
  In den Präsentationen von dem dritten Meilenstein haben einige Gruppen darüber berichtet, dass sie Junit Test mit Hilfe von Mockito erstellen. Aus diesem Grund haben wir uns Mockito ein wenig genauer angeschaut und uns dafür entschieden, dies bei einigen von unseren Tests auch zu verwenden. Mockito ist eine freie Programmbibliotek (bei unserem Programmierprojekt auf der Whitelist) um Mock-Objekte zu erstellen für Unit-Tests. Damit wir Mockito nutzen können mussten wir es natürlich auch ins Gradle einbinden.
- Diary
- QA
  Wir beginnen mit den QA Messungen, damit wir für den fünften Meilenstein genügend Informationen sammeln können.
- Chat Command
  Buttons im GUI hinzufügen, damit die verschiedenen Funktionen des Chats nicht nur per Command verfügbar sind. Playername wechseln und Ping abfragen sind nun per Knopfdruck möglich. Um eine Privatnachricht an einen anderen Player zu senden ist dies mit einem Doppelklick auf seinen Namen möglich

Eintrag vom 20. April 2018

Als Vorbereitung für den vierten Meilenstein sind wir alle Anforderungen durchgegangen und haben uns angeschaut was wir für diesen Meilenstein noch machen müssen.

- Committed
  Vor der Abgabe werden wir nochmals alle Dokumente und den Code überprüfen um sicherzustellen, dass wir nichts vergessen haben und alles auf dem neusten Stand sind.

- GUI
  Das Spiel wird bereits in einem GUI dargestellt und ist soweit wie möglich spielbar.

- Referee
  Wir müssen noch einen guten Weg finden um das Spiel zu Balancen und uns überlegen wie der Metropolenstatus erreicht werden kann.

- Unit-Tests
  Die Unittest haben wir bis jetzt ein wenig vernachlässigt, darum müssen wir jetzt noch einige Unittest schreiben mehr Unit-Tests schreiben. Als ersten müssen wir uns erkunden wie wir Singletons mit per Unittest überprüfen können, da wir einige Singletons in unserem Code verwenden.

- Victory
  Es muss einen Gewinner geben, wenn das Spiel bis zum Metropolenstatus gespielt wird und dieser muss in der Highscore ersichtlich sein.

- Chat Command
  Man muss nicht mit Chat Commands Funktionen des Games aufrufen. Alles ist im GUI machbar.

- QA
  Wir versuchen unser QA so wie geplant einzuhalten und die verschiedenen Punkte über die Zeit zu messen. Wir

Um alle diese Punkte zu erreichen müssen wir die gesamte Game Logic implementieren bzw. alle Gebäude und den Metropolenstatus implementieren damit jemand gewinnen kann. Bevor wir damit beginnen können müssen wir noch einiges an unserem Code verbessern und vereinfachen.
Wir versuchen diesen Meilenstein bereits einige der Bonuspunkte des fünften Meilensteins zu erledigen. Da wir bereits viel Vorarbeit für den vierten Meilenstein geleistet haben.

- Sound
  Ein Test-Sound ist bereits eingebunden. Unsere Playlist muss noch eingebunden werden und ausgewählt werden welches Lied wann gespielt werden soll.

- Full Screen

- Printing Proof Docs

- Mods

- Occupational Therapy
  Der Server kann bereits ohne Port gestartet werden, in diesem Fall wird der vordefinierte Port verwenden. Noch zu machen: Der Client soll auch mit dem vordefinierten Port starten, falls kein Port angegeben wird.

Milestone 3

Leider ist uns vor dem dritten Meilenstein ein Fehler unterlaufen und wir haben vergessen die Präsentation zu pushen. Aus diesem Grund haben wir uns für folgendes für den dritten und vierten Meilenstein vorgenommen: vor der Abgabe werden wir nochmals den gesamten Master Branch durchgehen und überprüfen ob alle Dokumente, in der neusten Version, vorhanden sind. Bei der Besprechung in der Übungsgruppe haben wir noch gutes Feedback bekommen. Wir werden unser Gradle anpassen damit wir die jar-Files aus dem Repro löschen können. Auch zu unserem QA haben wir noch zwei Tipps erhalten. Wir wollen unseren Code möglichst Modular aufbauen und wussten nicht wie wir dies Messen können. Dies wäre zum Beispiel messbar indem geschaut wird wie viele Klassen angepasst werden müssen um eine Textur für die Map zu ändern. Wir sollten nicht einfach für alle Klassen, die über 100 Zeilen haben, einen Unit Test schreiben, sondern für die Klassen mit wichtigen Funktionen. Wir haben bereits gute Vorarbeit für den Meilenstein 4 geleistet, da wir bereits für das gesamte Spiel ein GUI haben.

Eintrag vom 11. April 2018

Folgende Punkte zum Meilenstein 3 haben wir bereits fertiggestellt:

- Broadcast
- Commandline
- Chat GUI
- GameState ist auf dem Server
- Playerlist
- Gamelist
- Whisper

Folgende Punkte müssen wir noch abarbeiten:

- Protokoll muss aktualisiert werden
  Isabel HighscorePackge
  Jannik LobbyPackage
- Highscore
  Der Highscore funktioniert nur wenn das Spiel über die Commandline gestartet wird. Wird das Spiel mit der Jar-Datei gestartet funktioniert das Auslesen der txt-Datei nicht.
- Game ist spielbar
  Gebäude können auf der Karte gesetzt werden aber die GameLogic muss noch verknüpft werden.
- QA
  Pascal hat das QA mit Marco angeschaut und wir werden noch hinzufügen, dass die Methoden nicht länger als eine gewisse Anzahl von Zeilen sein soll.
- Library
  Log4J muss noch eingebunden werden.
- Lobby
  Lobby muss gestartet werden können:
  Button Create Lobby → neues Fenster, Map auswählen und Lobby erstellen oder gestartete Lobby im Hauptmenu auswählen und joinen.

Eintrag vom 10. April 2018

Heute arbeiten wir gemeinsam am Programmierprojekt weiter. Zu Beginn erklärt uns Pascal die Umstrukturierung der Projektstruktur und die Änderungen die er im Kommunikationsprotokoll vorgenommen hat. Die Priorität haben wie nach dem Feedback von Meilenstein 2 herausgelöscht und einige Verbesserungen vorgenommen.

- Package Types
  Sind in einer Enum Klasse definiert.
- Package
  Jades Package hat seine eigene Klasse:
  - ChatPackage, ConnectionPackage, PingPackage, UserPackage
  - Heute werden folgende Package hinzugefügt:
    * HighscorePackage (Isabel)
    * LobbyPackage (Jannik)

- HandlerManager
  Jedes Package hat Server- bzw. Client-Seitig einen Handler und einen Manager.

Wir haben in einigen Branches gleichzeitig gearbeitet. Nikolai und Pascal mergen alle Branches auf den Masterbranch und verknüpfen die verschiedenen Schnittstellen miteinander als Vorbereitung auf den Meilenstein 3.

Eintrag zum MapRendering

Ein Anspruch von uns an unser Spiel ist, dass es nicht nur 2D dargestellt wird, sondern in einem fake 3D bzw. in einer isometrischen Ansicht. Jannik hat sich innig damit beschäftigt und einen Prototypen in Swing erstellt. Siehe Abbildung 3
In einem zweidimensionalen Array wird jedes einzelne Quadrate (Tiles) umgerechnet. Für die Umrechnung haben wir die Methoden toIso() und toCartesian() geschrieben. Beim ersten Rendern der Map dauert es etwas länger da jedes Tile einzeln ausgerechnet werden muss. Als der Prototyp fertiggestellt war, haben wir uns Gedanken darüber gemacht wie wir die Struktur in unserem Spiel aufbauen wollen.

Unsere Idee:
Die Map wird auf dem Server und auf dem Client geladen. Auf dem Client muss die Map im Userinterface angezeigt werden. Auf dem Server benötigen wir nur die Tile Repräsentation damit wir die Tiles mit unserer Spiellogik verknüpfen können. Wenn ein Spieler etwas bauen will wird beim Server angefragt ob dies erlaubt ist oder nicht. Falls gebaut werden darf, wird das Gebäude auf dem Server gebaut und an alle Clients gesendet.

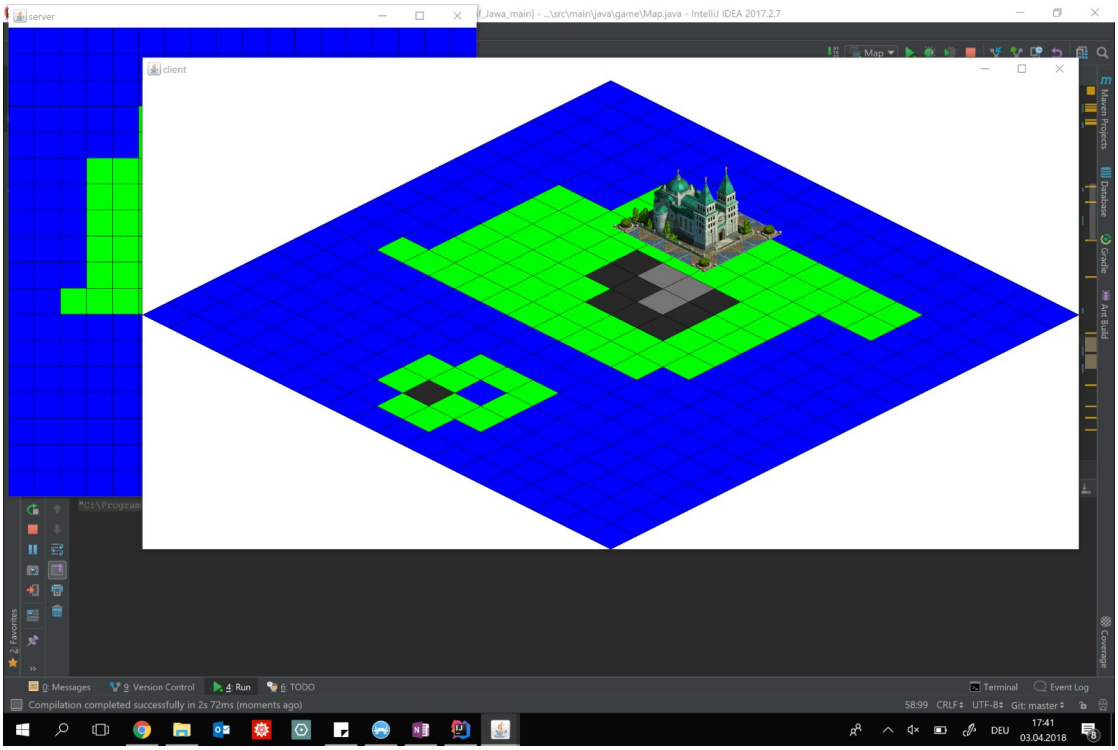

Abbildung 1: Testlauf der Map.

Alle Berechnungen und die gesamte Spiellogik spielen sich auf der "Kartesischen Karteäb und nur der Teilausschnitt der beim Spieler angezeigt wird, wird ins ïsometrische Kartensystemümgerechnet und im Userinterface angezeigt. Sonst hätten wir pro Sekunde 1 Billion Operation um die gesamte Map für jeden Spieler zu aktualisieren.
Im JavaFX haben wir zwei Canvas Elemente die gleich gross sind, zum einen die Game Canvas und die Heads-Up-Display Canvas. Wenn auf dem HUD Canvas kein Element verfügbar ist auf das geklickt werden kann wird man automatisch an das Game Canvas weitergeleitet. Im Game Canvas ist die gesamte Map gespeichert und auf dem HUD Canvas haben wir verschieden Elemente wie zum Beispiel die Minimap.

About the Game

Hauptziel: Aufbau einer Zivilisation mit Metropolenstatus.
Das Spiel ist in drei Phasen aufgeteilt:

- Insel aussuchen und Basis bauen
- Bau von Ressourcengebäuden (Minen, Farmen, etc.) um damit die entsprechenden Ressourcen zu sammeln

- Verbesserungen der Gebäude (Aufstieg in höhere Level) damit schneller Ressourcen gesammelt werden können

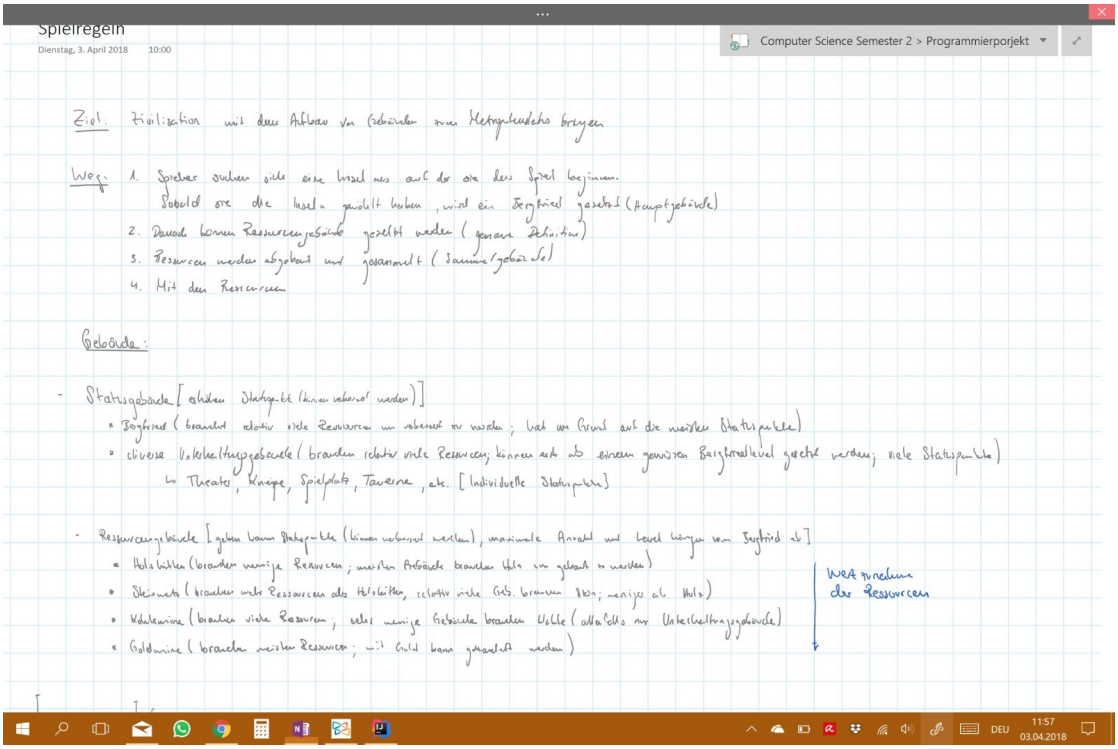

Abbildung 2: Erste Ideen von Nikolai 1

Aus der Diskussion was wir für den dritten Meilenstein zur basic GameLogic alles benötigen, hat sich folgendes ergeben. Es muss möglich sein eine Insel auszuwählen und eine Basis zu bauen. Sobald die Basis gebaut ist kann das erste Ressourcen-Gebäude gebaut werden. Um Steuern einzunehmen muss auch gleich das erste Einfamilienhaus gebaut werden.
Um die basic GameLogic zu zeigen benötigen wir daher nur ein Ressourcen-Gebäude, mit dem wir eine Ressource (z.B. Holz oder Stein) generieren können und Bewohner die Steuern bezahlen. Dies ist die Basis für das gesamte Spiel.
Zu einem späteren Zeitpunkt werden noch weitere Gebäudearten hinzugefügt. Dann können wir auch noch genau festlegen, wie wir die verschiedenen Level der Gebäude bzw. der Basis festlegen möchten.

Das Spiel kann beliebig erweitert werden, zum Beispiel:

- Ressourcengebäude:
  Supermärkte, Steinmetze, Goldminen, Eisen, Kohle, Kaffee, Diamanten, Wolle

- Spassgebäude:
  Theater, Taverne, Bordell, Thermen
- Bildungsbebäude:
  Uni Basel, Kirche, Mosche, Tempel, Opfergebäude, Altar
- Verwaltungsgebäude:
  Gericht, Gefängnis, Verwaltung

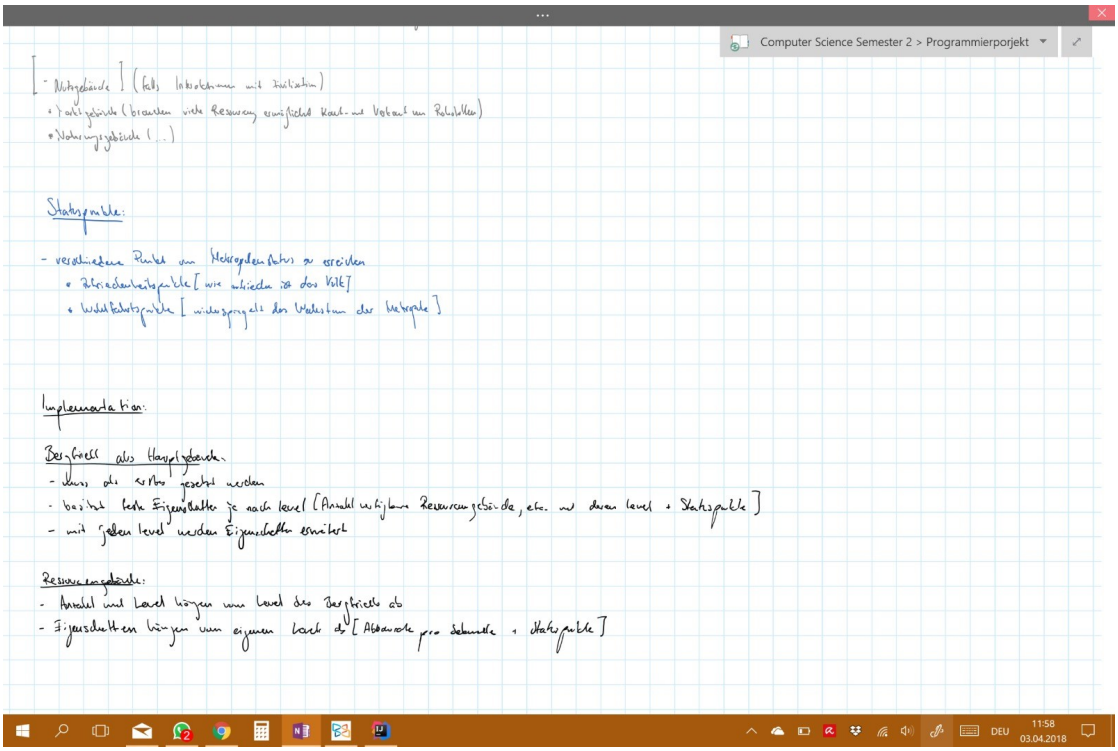

Abbildung 3: Erste Ideen von Nikolai 2

Eintrag vom 3. April 2018

Bevor am Programmierprojekt weiter gearbeitet haben sind wir die Punkte für den Meilenstein 3 durchgegangen und haben alles besprochen.

- About the Game
  Wir hatten bereits eine sehr gute Grundidee, diese wird nun von Nikolai ausgearbeitet und in diesem Diary Eintrag angehängt. Sobald die Ideen ausgearbeitet sind werden wir diese Ideen gemeinsam durchgehen und besprechen.

- Broadcast
  Jannik wird bei Marco nachfragen wie dieser Punkt zu verstehen ist. Sobald er eine Antwort erhält wird er alle updaten.

- Build Script
  Die ausführbare Jar-Datei ist bereits funktionsfähig. Im nächsten Schritt werden wir das Erstellen des Java Docs im Gradle einbinden, damit dies automatisiert wird.

- Commandline
  Jannik wird den Code anpasse, dass das Starten des Spiels mit den vorgegebenen Parametern funktioniert.
  $(client < hostadress >:< port > [< username >]|server < port >)$

- Chat im GUI
  Isabel hat bereits einen Entwurf des Userinterface für den Chat erstellt. Heute werden Pascal und Isabel das Userinterface mit dem Code verknüpfen.

- Gamelist
  Im Userinterface soll eine Liste mit allen Lobbies erstellt werden. Bei jeder Lobbie wird der aktuelle Status waiting, running, paused oder finished angezeigt. Die Eigenschaften einer Enum Klasse kommen uns hier sehr entgegen, da wir nur diese 4 verschiedenen Status verwenden wollen.

- GameLogic
  Für diesen Meilenstein soll die Game Logic in einer einfachen Form bereits implementiert und spielbar sein. Damit wir unsere Game Logic im Spiel umsetzten können müssen wir zuerst das Rendering der Map und das Entity System fertigstellen. Um die Umsetzung der Game Logic kümmern wir uns in der Kalenderwoche 15.

- GameState
  Der Server leitet alles an den Client weiter, es sollen keine Daten die das Spielgeschehen beeinflussen auf dem Client sein, damit der Client nicht cheaten kann.

- Highscore
  Im Userinterface soll der Highscore angezeigt werden. Der Highscore muss in einem File gespeichert werden, damit dieser auch nach einem Neustart des Spiels noch sichtbar ist. Im Highscore soll der Spielname, der Gewinner, die zu erreichende Punktzahl und die benötigte Zeit angezeigt werden. So haben wir später die Möglichkeit verschiedene Spielmodi zu implementieren. Zum Beispiel eine Speed Runde in der nur 200 Punkte für den Metropolenstatus benötigt werden.

- Library
  Pascal wird nachfragen ob JavaFX als Library gezahlt wird. Falls JavaFX keine Library ist werden wir Log4J in unserem Spiel hinzufügen. Damit Error Logs schön

angezeigt und gespeichert werden können. Dies vereinfacht und das Debugging.
→ JavaFx zählt nicht als Library.

- Manual
  Das Manual werden wir erst weiterschreiben sobald unser Userinterface und die Game Logic fertig sind. Damit wir diese nicht ständig anpassen müssen.

- Playerlist
  Im Userinterface soll eine Liste der Spieler die online sind angezeigt werden. Sobald ein Spieler seinen Namen ändert muss dieser auch in der Spielerliste geändert werden. Sollten wir noch genügend Zeit haben möchten wir noch einen Rechtsklick hinzufügen um eine Direktnachricht an den jeweiligen User zu senden.

- Protocol
  Alle neuen Packages müssen im Protokoll fortlaufend dokumentiert werden.

Milestone 2

Wir haben den zweiten Meilensteil erfolgreich gemeistert und unsere Ziele grösstenteils erreicht. Für den nächsten Meilenstein haben wir einige Dinge, die wir anders handhaben müssen. Alle Tagebucheinträge werden zusätzlich zur Webseite auch in einem PDF gespeichert und auf den master branch gepusht. In unserem Protokoll werden wir die Priority löschen. Wir benötigen die Priorität nicht und es dauert länger, die Pakete nach Priority zu sortieren als einfach abzuarbeiten. Als Beispiel für ein Packet werden wir die Chain im Wiki genauer beschreiben. User Quality Assurance muss komplett überarbeitet werden. Es müssen klar messbare Ziele definiert werden. Wir werden uns auch für die Arbeiten von Meilenstein 3 mindestens dreimal pro Woche treffen und gemeinsam am Projekt weiterarbeiten. Der ganze Dienstag ist für das Programmierprojekt reserviert und alle Teammitglieder sind anwesend.

Eintrag vom 25. März 2018

Am Abend vor dem zweiten Meilenstein trafen wir uns nocheinmal um die Anforderungen ein letztes Mal durch zu gehen.

- Quality Assurance
  Alle Informationen zu unserer Qualitätssicherung werden auf der Webseite eingebunden.

- Protokoll
  Die Dokumentation des Protokolls wurde fertiggestellt.

- Diary
  Alle lokalen Tagebucheinträge wurden online gestellt.

- Logout
  Der Client wird nun aus dem Server ausgetragen sobald er sich ausloggt oder die Verbindung verliert.

- Console Logs
  Alle ßystem out prints"werden durch Console Logs ersetzt.

Eintrag vom 23. März 2018

Einige Arbeiten für den zweiten Meilenstein konnten am Dienstag noch nicht fertiggestellt werden. Aus diesem Grund haben wir uns am Freitag nochmals getroffen.

- Ping/Pong
  Erste Implementierung des Ping. Ein Ping wird gesendet sobald das Programm gestartet wird.

- Nickname
  Beim Starten des Clients wird der Nickname aus dem Betriebssystem ausgelesen und vorgeschlagen. Der Nickname kann mit dem Befehl
  nick geändert werden.

- Whisper
  Mit dem Befehl wisper ïicknameẗkann eine direkte Nachricht an einen anderen Client versendet werden

Eintrag vom 20. März 2018

Gemeinsam haben wir uns die Anforderungen für den Meilenstein 2 angeschaut und mit unserem aktuellen Stand abgeglichen.

- Source Code
  Sobald die Funktionen für den zweiten Meilenstein implementiert sind, wird der gesamte Code nochmals mit dem Style Check überprüft und die fehlenden Kommentare hinzugefügt.

- Nickname
  Der Nickname kann bereits frei gewählt werden. Ist der Name bereits vergeben so werden Zahlen startend bei 01 am Ende eingefügt. Für den Nickname müssen noch zwei Funktionen eingebaut werden. (Username auslesen, ändern)

- Chat
  Nach diesem Meeting überlegen wir uns gemeinsam, wie wir den Chat implementieren können.

- Login
  Die Verbindung vom Client auf den Server funktioniert bereits.

- Logout
  Beim Logout müssen zwei verschiedene Situationen beachtet werden (lost connection, logged out).
- Protokoll
  Das Protokoll wird auf Wiki Dokumentiert und auf der Webseite verlinkt.
- PingPong
  Muss noch implementiert werden.

Eintrag vom 13. März 2018

Kurze Sitzung, Freitag, 02.09.18
Übers Wochenende werden wir uns alle mit dem Aufsetzen der Server/Client Struktur und dem Protokoll befassen, damit wir am Dienstag alle auf dem gleichen Stand sind und gemeinsam weiterarbeiten können.

Arbeiten am Programmierprojekt, Dienstag, 13.03.18
Wir haben jeweils den gesamten Dienstag eingeplant um gemeinsam am Programmierprojekt zu arbeiten. Heute arbeiten Pascal und Nikolai am Protokoll, Jannik an der Server/Client Struktur und Isabel am User Interface.

Milestone 1

Nicht ganz eine Woche hatten wir Zeit um eine Spielidee auszuarbeiten. Nach vielen Stunden Arbeit haben wir sowohl die nötigen Programme installiert und Informationen zusammengetragen, wie auch viele unserer Ideen wieder verworfen und sind nun bereit für den ersten Meilenstein. Die wichtigsten Punkte unserer „Meetings“ werden wir hier in Form eines Tagebuchs festhalten.

Was bisher geschah:
Unsere Spielidee ist es, auf einer Insel eine Metropole zu errichten. Nicht auf allen Inseln sind die Rohstoffe in gleicher Menge vorhanden, darum sollte die Insel, bevor es los geht, klug ausgewählt werden. Sobald man sich für eine Insel entschieden hat, startet das Echtzeit Strategie-Spiel. Die SpielerInnen sammeln und tauschen Rohstoffe um ihr Ziel (den Metropolen Status) möglichst schnell zu erreichen. Währenddessen muss auch darauf geachtet werden, dass man nicht unter den festgelegten Threshold fällt. Sollte dies doch geschehen, so hat derjenige das Spiel leider verloren
Ausgangslage: Dieses Projekt wird mit der Programmiersprache Java erstellt. Das Ziel des Spiels ist es, auf einer Insel eine Metropole zu errichten. Dies war unsere Ausgangslage, als wir uns Gedanken zum Namen unseres Spieles machten. Ein kurzer Kontrollblick auf die Karte zeigte, dass eine indonesische Insel tatsächlich auch „Jawa“ heisst. Dies mussten wir für den Namen unseres Spiels ausnutzen. Der Spieler oder die Spielerin die es schafft als erstes eine Metropole aufzubauen wird somit zum "King of Jawa".

Nebst der Spielidee haben wir uns auch mit dem Netzwerk beschäftigt. Genauer gesagt, wie die Server – Client Struktur aufgebaut werden muss. Damit die SpielerInnen sich auf der Karte bewegen können, muss eine Anfrage an den Server gesendet und dort validiert werden. Dasselbe gilt für Tauschgeschäfte unter den SpielerInnen, aber auch um die Metropole auf zu bauen. Das bedeutet, dass der Client bzw. der Spieler nur Spielzüge durchführen kann, welche vom Server validiert wurden.
Der Projektplan darf natürlich auch nicht fehlen. Darin haben wir versucht die Arbeit unter uns so einzuteilen, dass jeder von uns während des gesamten Projekts etwas zu tun hat. Wie allgemein bekannt ist, lässt es sich einfacher Arbeiten, wenn man sich für die gestellte Aufgabe interessiert. Auch dies haben wir in die Projektplanung mit einbezogen. Nach nur knapp einer Woche ist es sehr schwierig abzuschätzen, was alles auf uns zukommen wird. Aus diesem Grund werden wir den Projektplan laufend erweitern und falls nötig anpassen.
Der Abgabetermin für den ersten Meilenstein ist in weniger als einem Tag. Morgen werden wir die Präsentation noch einmal durchgehen, damit wir den ersten Meilenstein gut vorbereitet hinter uns bringen können.