Terror Nightmare Towers (TNT)** - ***Santorini*** mit einem Zombie/Horror Thema

	10.05.23	15.05.	21.05.	28.05.	04.06.	11.06.	18.06.	25.06.	02.07.	09.07.	16.07.	23.07.	30.07.	02.08. – 12:00
Finn	Planung	Logik & Testing												
Tim		Maussteuerung												
Youran		Spieler												
Fabio		GUI & Models												

Release I
* Das Team ist im QIS angemeldet
* Es existiert ein Projekt im Gitlab (Fork vom Template)
* Alle Teammitglieder sind im Gitlab in eurem Projekt Mitglieder
* Die Anforderungen sind aufgenommen und dokumentiert (Story Cards, etc.)
* Es gibt einen Projektplan (Arbeitspakete, Zeitplanung, Gantt-Chart, Meilensteine, etc.)
* Es gibt eine Verantwortlichkeitsverteilung im Team und diese ist schriftlich festgehalten.
Release II
* Es existiert ein erkennbares minimales Spiel
* Das Spiel kann von 2 menschlichen Spielern gespielt werden
* Das Spiel ist "per Hand" spielbar, d.H. Regeleinhaltung wird noch nicht unbedingt vom Computer überprüft
* Es existiert eine GUI und die Steuerung des Spiels mit der Maus funktioniert
* Es gibt einen "Open Issues Report" in dem noch zu erledigende Arbeiten, Bugs, etc. dargestellt sind.
Release III
* Das vollständige Spiel inklusiver aller Pflichtfeatures, ausreichend Wahlfeatures und Zusatzfeatures ist implementiert, getestet und spielbar.
* Das gesamte Projekt muss inklusive aller Doku/Artefakte/Anleitungen/etc. im Gitlab abgegeben werden.
* 20 Minütige Abschusspräsentation

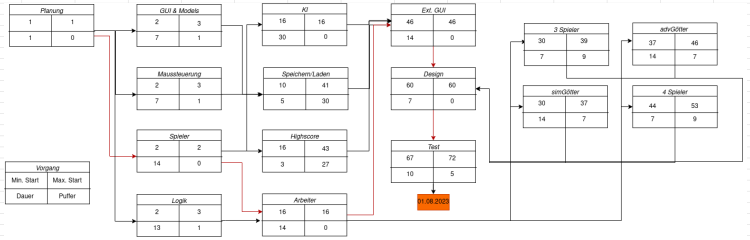
1. Die GUI passt Thematisch zu dem Zombie/Horror-Theme des Spiels.
2. Zu Programmstart kann gewählt werden, wie viele Spieler am Spiel teilnehmen (wenn als Wahlfeature implementiert) und welcher Spieler von dem Computer oder einem Menschen gespielt wird.
3. Die GUI zeigt das Spielfeld in der Draufsicht (Thema des Spiels beachten). Die Figuren und Bauwerke sind durch sinnvolle Symbole oder Grafiken dargestellt.
4. Das Spiel ist mit der Maus spielbar.
5. Das Spiel zeigt den aktiven Spieler an bzw. highlightet die eigenen Figuren
6. Ungültige Zugversuche werden ignoriert.
7. Das Programm endet, wenn ein Spieler gewinnt, bzw. alle anderen verlieren.
8. Am Ende des Spiels wird der Gewinner hervorgehoben.
9. Es kann jederzeit zum Auswahlbildschirm des Spiels zurückgekehrt werden und von dort aus ein neues Spiel begonnen werden.

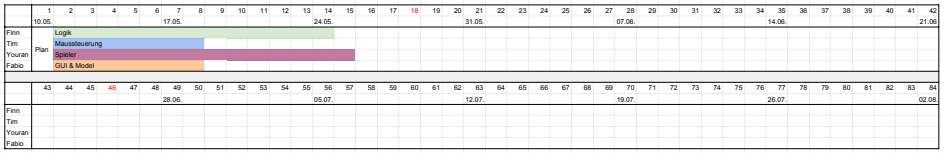
3 Spieler Variante (5P.)
4 Spieler Variante (5P.)
Menüführung (1P.)
Speichern/Laden (2P.)
Mehrsprachigkeit (1P.)
Resizeable GUI (2P.)
Spielernamen und -farben (1P.)
High Score (2P.)
Wählbares Terror Theme (3P.)
Anzahl der Bausteine wählbar (2P.)
Kompetente KI (3P.)
Anzahl der Arbeiter wählbar (3P.)
Größe des Spielfeldes Wählbar (2P.)
Die Welt ist eine "Kugel" (3P.)
Macht der Götter Simple (1P. je 2 Götterkarten, Max 5 Punkte)
Macht der Götter Advanced (1P. je 4 Götterkarten, Max 5 Punkte)

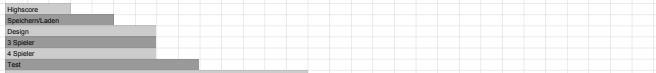
Codequalität
Testabdeckung
Dokumentation
Beteiligung in Moodleoverflows
Anforderungsdokumentation
Architekturdokumentation
Javadoc-Dokumentation
Codingstyle, Metriken, Testabdeckung

Das Spiel kann von 2 menschlichen und/oder Computer-Spielern gespielt werden.
Das Spielfeld wird dargestellt.
Figuren können ausgewählt und den Spielregeln entsprechend bewegt werden.
Anschließend kann ein Gebäude(teil) den Regeln entsprechend errichtet werden.
Das Spiel endet den Regeln entsprechend.
Am Ende des Spiels wird der Sieger ermittelt und den Spielern angezeigt.

Bewertungskriterien
1. **Produktbezogene Qualitätsaspekte (50 P.)**
   1. Funktionalität und Zuverlässigkeit (30 P.)
      1. Implementierung aller Pflichtfeatures (14 P.)
      2. Implementierung ausgewählter Wahlfeatures (6 P.)
      3. Fehlerfreiheit der implementierten Funktionen (10 P.)
   2. Gebrauchstauglichkeit (15 P.)
      1. Verständlichkeit, Erlernbarkeit, Bedienbarkeit und Attraktivität des Produkts
      2. Abfangen von Bedienfehlern
   3. Effizienz/Performanz (5 P.)
      1. Keine unverhältnismäßig lange Rechen-/Antwortzeit
      2. Angemessener Ressourcenverbrauch (Arbeitsspeicher etc.)
2. **Entwicklungsbezogene Qualitätsaspekte (30 P.)**
   1. Stabile und modifizierbare Architektur (10 P.)
   2. Testing (8 P.)
      1. Testqualität
      2. Prozentuale Testabdeckung
   3. Einhaltung Metriken (4 P.)
   4. Codingstyle (4 P.)
      1. Einhaltung Codierungskonventionen
      2. Verständlicher Code (sprechende Var.-/Methodennamen, Kommentare)
      3. Einfacher Code (wo möglich)
   5. Versionsverwaltung (4 P.)
      1. Sprechende Commit-Nachrichten
      2. Sinnvolle Commitgrößen
3. **Dokumentation (20 P.)**
   1. Lückenlose Dokumentation der Anforderungen (Usecase-Diagramm, Storycards) (3 P.)
   2. Qualitativ hochwertige Dokumentation der Architektur inkl. Begründung (7 P.)
   3. Bedienungsanleitung für das Produkt (2 P.)
   4. Javadoc-Dokumentation (4 P.)
   5. Abschlusspräsentation (4 P.)







Anforderungen

Im diesjährigen Software-Engineering Praktikum soll das PC-Spiel **Terror Nightmare Towers (TNT)** entwickelt werden.

Entwickelt werden soll ein PC-Spiel basierend auf dem Regelwerk des Brettspiels ***Santorini*** mit einem Zombie/Horror Thema. Das Regelwerk des Brettspiels ist im Moodle zu finden.

Bei dem Spiel soll es möglich sein, gegeneinander als auch gegen den Computer zu spielen.

Das Spiel soll dafür über eine grafische Benutzerschnittstelle verfügen.

Dafür müssen eine Reihe von **Pflicht-**, **Wahl-** und **Zusatzfeatures** implementiert werden (genaue Beschreibung weiter unten). Das Spiel soll in englischer Sprache umgesetzt werden. Die Entwicklung soll sich dabei in drei Releases gliedern.

Ab Release 2 muss ein auslieferbares Produkt existieren. Dies umfasst neben einem lauffähigen Programm auch eine vollständige Dokumentation von Programm und Code, als auch hinreichende Abdeckung der Programmfunktionalität durch Tests.

Im Folgenden werden die Pflicht und Wahlfeatures näher erläutert.

Pflichtfeatures

Grundlegende Spielfunktionalitäten

Das Spiel kann von 2 menschlichen und/oder Computer-Spielern gespielt werden.

Das Spielfeld wird dargestellt.

Figuren können ausgewählt und den Spielregeln entsprechend bewegt werden. Anschließend kann ein Gebäude(teil) den Regeln entsprechend errichtet werden.

Das Spiel endet den Regeln entsprechend.

Am Ende des Spiels wird der Sieger ermittelt und den Spielern angezeigt.

GUI-basiertes Frontend

1. Die GUI passt Thematisch zu dem Zombie/Horror-Theme des Spiels.
2. Zu Programmstart kann gewählt werden, wie viele Spieler am Spiel teilnehmen (wenn als Wahlfeature implementiert) und welcher Spieler von dem Computer oder einem Menschen gespielt wird.
3. Die GUI zeigt das Spielfeld in der Draufsicht (Thema des Spiels beachten). Die Figuren und Bauwerke sind durch sinnvolle Symbole oder Grafiken dargestellt.
4. Das Spiel ist mit der Maus spielbar.
5. Das Spiel zeigt den aktiven Spieler an bzw. highlightet die eigenen Figuren
6. Ungültige Zugversuche werden ignoriert.
7. Das Programm endet, wenn ein Spieler gewinnt, bzw. alle anderen verlieren.
8. Am Ende des Spiels wird der Gewinner hervorgehoben.
9. Es kann jederzeit zum Auswahlbildschirm des Spiels zurückgekehrt werden und von dort aus ein neues Spiel begonnen werden.

Spiel gegen den Computer

Für das Spiel gegen den Computer, soll der Computer aus allen Zügen, die er aktuell ausführen kann, den wählen, der aktuell am sinnvollsten erscheint (Greedy Strategie o.Ä.).

Spiel gegen Menschliche Spieler

Die menschlichen Spieler wechseln sich Reihum an einem PC ab (Hot-Seat). Nach dem Zug eines Spielers zeigt das Programm an, wer als Nächstes an der Reihe ist.

Wahlfeatures

3 Spieler Variante (5P.)

Zum Start des Spiels kann neben der Auswahl für menschliche und Computerspieler auch die Anzahl der Spieler eingestellt werden. Die 3 Spieler spielen ansonsten analog zu der 2 Spieler Variante.
(anders als in den offiziellen Regeln sind die Götterkarten hierfür keine Voraussetzung)

4 Spieler Variante (5P.)

Zum Start des Spiels kann neben der Auswahl für menschliche und Computerspieler auch die Anzahl der Spieler eingestellt werden. Die 4 Spieler spielen wie in den Regeln dargestellt in Zer-Teams. Die Teams gewinnen gemeinsam, bzw. verlieren gemeinsam, sobald ein Teammitglied verliert.

Menüführung (1P.)

Nach dem Starten des Programms öffnet sich ein "Hauptmenü" in dem sich alle Untermenüs. Einstellungsoptionen, Möglichkeit ein spiel zu starten oder das Programm zu beenden befinden. Das Menü kann außerdem jederzeit aus dem laufenden Spiel aufgerufen und wieder geschlossen werden, ohne dass das

Allgemeine Informationen zur Durchführung

Im Software-Engineering Praktikum soll ein PC-Spiel basierend auf einem Brettspiel entwickelt werden. Dabei sollen die aus der Vorlesung bekannten Prinzipien zur Entwicklung von qualitativ hochwertiger Software beachtet werden.

Die Entwicklung findet in Dreier- oder Vierergruppen statt. Die Umsetzung des Projekts findet in der Zeit vom 26.04.2023 bis zum 02.08.2023 statt.

Für das Praktikum muss man sich im Uni **QIS** anmelden.
Die Deadline zur Anmeldung ist der ***15.05. -- 23:59 Uhr**. Diese Anmeldung ist verpflichtend. Wer sich bis zur Deadline nicht angemeldet hat, kann nicht weiter am Praktikum teilnehmen.

Umsetzung und Infrastruktur

Die Programmierung erfolgt ausschließlich in der Sprache Java, Version 11. Neben JavaFX (komplett) werden keine weiteren externen Libraries benötigt. Sollten ihr weitere externe Libraries benutzen wollen, so ist das erlaubt. Ihr seid dann jedoch dafür selbst verantwortlich, dass diese vernünftig im Build Prozess eingebunden werden, sodass es ohne manuelle Nachinstallationen auf unseren Testsystemen läuft. Ihr könnt annehmen, dass außer Java 11 und JavaFX keine weiteren Pakete installiert sind.

Die Versionsverwaltung muss mit git erfolgen.

Tutorial

Für eine Einarbeitung in die im Praktikum verwendeten Tools steht ein [Tutorial](https://projects.isp.uni-luebeck.de/software-engineering-praktikum-dokumente/tutorial) zur Verfügung. Am 28.04. findet zudem ein Einführungsworkshop in die im Praktikum verwendeten Tools durch das Unternehmen Capgemini statt. Eine Teilnahme ist nicht verpflichtend, wird aber empfohlen. Details zu der Veranstaltung werden rechtzeitig im Moodle bekannt gegeben.

Infrastruktur

Für die Umsetzung des Projekts steht folgende Infrastruktur zur Verfügung:

* [Gitlab](https://projects.isp.uni-luebeck.de/) zur Verwaltung des Source-Codes.

Falls ein Teilnehmer noch über keinen Account in den genannten Plattformen verfügt, kann dieser auf den jeweiligen Seiten selbstständig angelegt werden. [Registrieren Sie sich dazu einen Account an unserem Gitlab](https://projects.isp.uni-luebeck.de/users/sign_up): Nutzen Sie dazu Ihre offizielle E-Mail-Adresse an der Uni-Lübeck als E-Mail-Adresse und den lokalen Teil dieser E-Mail-Adresse als Benutzername. Wenn Sie also die E-Mail-Adresse `erika.mustermann@student.uni-luebeck.de` haben, registrieren Sie sich mit dem Benutzernamen `erika.mustermann`. Teilnehmer der Vorlesung im letzten Semester haben bereits einen Account in diesem Gitlab und können diesen weiter nutzen.

Entwicklungsumgebung

Für die Entwicklung ist keine spezifische Entwicklungsumgebung festgelegt. Wir empfehlen die Verwendung einer bekannten IDE wie Eclipse, IntelliJ IDEA oder Netbeans, da für diese IDEs in der Regel viele Erweiterungen und Dokumentation verfügbar sind. Das Build-Management muss in jedem Fall via Maven erfolgen.

Tutoren und Präsenzzeiten

Die Tutoren stehen zur persönlichen Betreuung Mittwochs von 12 bis 18 für euch bereit. Auch allgemeine Fragen, insbesondere zu organisatorischen Themen können dort besprochen werden.

Ausscheiden aus dem Praktikum

Sollte ein Gruppenmitglied auf eigenen Wunsch aus dem Praktikum ausscheiden werden je nach Ausstiegszeit die Anforderungen an die Gruppe individuell angepasst. In diesem Fall nehmen die Gruppenmitglieder bitte unmittelbar Kontakt mit Ludwig Pechmann und Philipp Bende auf. Ebenso ist diesen das Ausscheiden durch das Gruppenmitglied unverzüglich mitzuteilen.

Sollte ein Gruppenmitglied offenkundig nicht willens sein sich in das Praktikum einzubringen, kann das Mitglied aus dem Praktikum ausgeschlossen werden. In diesem Fall können sich die weiteren Gruppenmitglieder an Ludwig Pechmann und Philipp Bende wenden. Auch in diesem Fall können die Anforderungen individuell für die verbleibende Gruppe angepasst werden.

Ein Ausschluss oder Ausscheiden aus dem Praktikum wird als Nicht-Bestehen gewertet und entsprechend an das Prüfungsamt gemeldet.

Abgabe

Im Praktikum finden 3 Releases statt, bei denen jeweils der bisherige Stand des Projekts im Gitlab "abgegeben" und bewertet wird.

Zwischenabgaben

Release I
Deadline
28.05. -- 23:59

* Das Team ist im QIS angemeldet

laufende Spiel dadurch beendet wird.

Speichern/Laden (2P.)

Das aktuelle Spielfeld inklusive Kartenreihen, Handkarten, Ablagestapel, aktivem Spieler und Punkten kann in einer Datei gespeichert oder daraus geladen werden. Bezüglich des Speicherformats gibt es keine Vorgaben.

Mehrsprachigkeit (1P.)

Von jeder Stelle im Spiel aus kann zwischen zwei oder mehr Sprachen gewechselt werden (z.B. über das Menü).

Resizeable GUI (2P.)

Das GUI-basierte Frontend lässt sich jederzeit auf eine beliebige Größe ziehen.

Spielernamen und -farben (1P.)

Jedem Spieler (Mensch und Computer) kann ein Name und eine Farbe gegeben werden, die im Spiel angezeigt werden.

High Score (2P.)

Die Anzahl benötigter Züge des Gewinners jeder Partie wird mit den aktuellen High Scores verglichen und, wenn ausreichend hoch, in die Bestenliste eingetragen. In der Bestenliste befinden sich die 3 besten Ergebnisse bestehend aus Zuganzahl, Spielernamen, und ob es sich um einen Menschlichen oder Computerspieler handelte. Der High-Score ist persistent über das Beenden des Spiels.

Wählbares Terror Theme (3P.)

Es gibt wenigstens 2 Unterschiedliche grafische Themen (z.B. Untote, alles Blutig, etc.). Zwischen diesen Themen lässt sich vor und während des Spiels das Aussehen des Spiels verändern (z.B. über das Menü).

Anzahl der Bausteine wählbar (2P.)

Vor dem Spiel kann ausgewählt werden, wie viele Bausteine (Kuppel und Level 1-3) es gibt. Das Spiel kann demnach früher/später dadurch enden, dass ein Spieler nicht mehr Bauen kann.

Kompetente KI (3P.)

Bei der Wahl eines Computerspielers kann man dessen Spielstärke einstellen. Ein starker Computerspieler sollte versuchen möglichst schnell zu gewinnen, wenn er der Meinung ist, vor allen anderen Mitspielern gewinnen zu können, und ansonsten versuchen die anderen Spieler zu behindern.

Anzahl der Arbeiter wählbar (3P.)

Vor Spielstart kann ausgewählt werden, wie viele Arbeiter jeder Spieler zur Verfügung hat.

Größe des Spielfeldes Wählbar (2P.)

Die Größe des Spielfeldes kann vor Spielbeginn festgelegt werden.

Die Welt ist eine "Kugel" (3P.)

Arbeiter können das Spielfeld am Rand verlassen und dafür das entsprechende Feld am gegenüberliegenden Rand betreten.

Macht der Götter Simple (1P. je 2 Götterkarten, Max 5 Punkte)

Die Spielvariante "Macht der Götter" kann den Regeln entsprechend gewählt werden. Je 2 vollständig und richtig implementierter "Simple Gods" Götter gibt es einen Punkt.

Macht der Götter Advanced (1P. je 4 Götterkarten, Max 5 Punkte)

Die Spielvariante "Macht der Götter" kann den Regeln entsprechend gewählt werden. Je 4 vollständig und richtig implementierter "Advanced Gods" Götter gibt es einen Punkt.

Im diesjährigen Software-Engineering Praktikum soll das PC-Spiel **"Terror Nightmare Towers (TNT)"** entwickelt werden.

Entwickelt werden soll ein PC-Spiel basierend auf dem Regelwerk des Brettspiels **"Santorini"** mit einem Zombie/Horror Thema. Das Regelwerk des Brettspiels ist im Moodle zu finden.

Bei dem Spiel soll es möglich sein, gegeneinander als auch gegen den Computer zu spielen.

Das Spiel soll dafür über eine grafische Benutzerschnittstelle verfügen.

Dafür müssen eine Reihe von **Pflicht-**, **Wahl-** und **Zusatzfeatures** implementiert werden (genaue Beschreibung weiter unten). Das Spiel soll in englischer Sprache umgesetzt werden. Die Entwicklung soll sich dabei in drei Releases gliedern.

Ab Release 2 muss ein auslieferbares Produkt existieren. Dies umfasst neben einem lauffähigen Programm auch eine vollständige Dokumentation von Programm und Code, als auch hinreichende Abdeckung der Programmfunktionalität durch Tests.

Im Folgenden werden die Pflicht und Wahlfeatures näher erläutert.

Pflichtfeatures

Grundlegende Spielfunktionalitäten

Das Spiel kann von 2 menschlichen und/oder Computer-Spielern gespielt werden.

Das Spielfeld wird dargestellt.

Figuren können ausgewählt und den Spielregeln entsprechend bewegt werden. Anschließend kann ein Gebäude(teil) den Regeln entsprechend errichtet werden.

Das Spiel endet den Regeln entsprechend.

Am Ende des Spiels wird der Sieger ermittelt und den Spielern angezeigt.

GUI-basiertes Frontend

1. Die GUI passt Thematisch zu dem Zombie/Horror-Theme des Spiels.
2. Zu Programmstart kann gewählt werden, wie viele Spieler am Spiel teilnehmen (wenn als Wahlfeature implementiert) und welcher Spieler von dem Computer oder einem Menschen gespielt wird.
3. Die GUI zeigt das Spielfeld in der Draufsicht (Thema des Spiels beachten). Die Figuren und Bauwerke sind durch sinnvolle Symbole oder Grafiken dargestellt.
4. Das Spiel ist mit der Maus spielbar.
5. Das Spiel zeigt den aktiven Spieler an bzw. highlightet die eigenen Figuren
6. Ungültige Zugversuche werden ignoriert.
7. Das Programm endet, wenn ein Spieler gewinnt, bzw. alle anderen verlieren.
8. Am Ende des Spielt wird der Gewinner hervorgehoben.
9. Es kann jederzeit zum Auswahlbildschirm des Spiels zurückgekehrt werden und von dort aus ein neues Spiel begonnen werden.

Spiel gegen den Computer

Für das Spiel gegen den Computer, soll der Computer aus allen Zügen, die er aktuell ausführen kann, den wählen, der aktuell am sinnvollsten erscheint (Greedy Strategie o.Ä.).

Spiel gegen Menschliche Spieler

Die menschlichen Spieler wechseln sich Reihum an einem PC ab (Hot-Seat). Nach dem Zug eines Spielers zeigt das Programm an, wer als Nächstes an der Reihe ist.

Wahlfeatures

3 Spieler Variante (5P.)

Zum Start des Spiels kann neben der Auswahl für menschliche und Computerspieler auch die Anzahl der Spieler eingestellt werden. Die 3 Spieler spielen ansonsten analog zu der 2 Spieler Variante.
(anders als in den offiziellen Regeln sind die Götterkarten hierfür keine Voraussetzung)

4 Spieler Variante (5P.)

Zum Start des Spiels kann neben der Auswahl für menschliche und Computerspieler auch die Anzahl der Spieler eingestellt werden. Die 4 Spieler spielen wie in den Regeln dargestellt in Zer-Teams. Die Teams gewinnen gemeinsam, bzw. verlieren gemeinsam, sobald ein Teammitglied verliert.

Menüführung (1P.)

Nach dem Starten des Programms öffnet sich ein "Hauptmenü" in dem sich alle Untermenüs, Einstellungsoptionen, Möglichkeit ein spiel zu starten oder das Programm zu beenden befinden. Das Menü kann außerdem jederzeit aus dem laufenden Spiel aufgerufen und wieder geschlossen werden, ohne dass das laufende Spiel dadurch beendet wird.

Speichern/Laden (2P.)

Das aktuelle Spielfeld inklusive Kartenreihen, Handkarten, Ablagestapel, aktivem Spieler und Punkten kann in einer Datei gespeichert oder daraus geladen werden. Bezüglich des Speicherformats gibt es keine Vorgaben.

Mehrsprachigkeit (1P.)

Von jeder Stelle im Spiel aus kann zwischen zwei oder mehr Sprachen gewechselt werden (z.B. über das Menü).

Resizeable GUI (2P.)

Das GUI-basierte Frontend lässt sich jederzeit auf eine beliebige Größe ziehen.

Spielernamen und -farben (1P.)

Jedem Spieler (Mensch und Computer) kann ein Name und eine Farbe gegeben werden, die im Spiel angezeigt werden.

High Score (2P.)

Die Anzahl benötigter Züge des Gewinners jeder Partie wird mit den aktuellen High Scores verglichen und, wenn ausreichend hoch, in die Bestenliste eingetragen. In der Bestenliste befinden sich die 3 besten Ergebnisse bestehend aus Zuganzahl, Spielernamen, und ob es sich um einen Menschlichen oder Computerspieler handelte. Der High-Score ist persistent über das Beenden des Spiels.

* Es existiert ein Projekt im Gitlab (Fork vom Template)
* Alle Teammitglieder sind im Gitlab in eurem Projekt Mitglieder

* Die Anforderungen sind aufgenommen und dokumentiert (Story Cards, etc.)
* Es gibt einen Projektplan (Arbeitspakete, Zeitplanung, Gantt-Chart, Meilensteine, etc.)
* Es gibt eine Verantwortlichkeitsverteilung im Team und diese ist schriftlich festgehalten. Es könnte z.B. ein Teammitglied für Architektur, ein weiteres für Testing und ein drittes für die Projektleitung verantwortlich sein. Diese Verantwortlichkeiten bedeuten allerdings nicht, dass man seine Verantwortlichkeit alleine erledigen muss, sondern eher, dass man darauf zu achten hat, dass die jeweilige Aufgabe vom Team allgemein erfüllt wird.

Release II
Deadline
25.06. -- 23:59

* Es existiert ein erkennbares minimales Spiel
* Das Spiel kann von 2 menschlichen Spielern gespielt werden
* Das Spiel ist "per Hand" spielbar, d.H. Regeleinhaltung wird noch nicht unbedingt vom Computer überprüft
* Es existiert eine GUI und die Steuerung des Spiels mit der Maus funktioniert
* Es gibt einen "Open Issues Report" in dem noch zu erledigende Arbeiten, Bugs, etc. dargestellt sind.

Die oben genannten Zwischenabgaben fließen nicht in die Bewertung des Praktikums ein. Bei grober Nichterfüllung der Anforderungen wird jedoch eine Nachbesserung innerhalb der nächsten 10 Tage gefordert. Erfolgt diese nicht oder nicht hinreichend, scheidet die Gruppe vorzeitig aus dem Praktikum aus.

Wichtig: Die Zwischenabgaben dienen dazu einen Minimalfortschritt im Projekt sicherzustellen und euch auf mögliche Schwachstellen in eurem Projekt hinzuweisen. Ein Erreichen der Endabgabe garantiert weder ein Bestehen des Praktikums noch eine gute Note.

Endabgabe

Release III
Deadline
02.08. -- 12:00

* Das vollständige Spiel inklusiver aller Pflichtfeatures, ausreichend Wahlfeatures und Zusatzfeatures ist implementiert, getestet und spielbar.
* Das gesamte Projekt muss inklusive aller Doku/Artefakte/Anleitungen/etc. im Gitlab abgegeben werden.

* 20 Minütige Abschlusspräsentation

Abschlusspräsentation

In der Abschlusspräsentation soll das Produkt vorgestellt werden. Ihr habt dabei die Möglichkeit die Funktionalität zu demonstrieren und auf Besonderheiten eures Produkts hinzuweisen.

Abgabeformat

An den Schlusstagen der Releases, inklusive Endabgabe, muss der zu überprüfende/bewertende Stand mittels Tags (`r1`, `r2`, ...) im git-Repository gekennzeichnet werden. Zusätzliche Dokumente (insbesondere die vollständige Dokumentation) sind als PDF-Dokumente `Bedienungsanleitung.pdf`, `Anforderungsanalyse.pdf` und `Architektur.pdf` im Hauptordner des git-Repositories zur Verfügung zu stellen.

Details zur Umsetzung der Anforderungen

Für das zu implementierende Spiel gelten die Regeln in der pdf-Datei "Regelwerk" im Moodle. Im Praktikum müssen eine Reihe von Features umgesetzt werden. Diese gliedern sich in Pflicht- und Wahl und Zusatzfeatures.
Die genaue Beschreibung findet sich unter [Features](https://projects.isp.uni-luebeck.de/sweng-prak-2023/aufgabenstellung/-/blob/master/Software%20Entwicklung/Software%20Entwicklung.md)

Pflichtfeatures müssen in jedem Fall komplett umgesetzt werden.

Wahlfeatures sind jeweils mit einer Punktezahl versehen. Es müssen Wahlfeatures im Wert von 30 Punkten umgesetzt werden. Umsetzen von mehr als 30 Punkten kann zu Bonuspunkten führen. Die korrekte Umsetzung der Features hat jedoch einen höheren Stellenwert als die Anzahl umgesetzter Features.

Zusatzfeatures werden im Laufe des Praktikums (nach Release II) bekannt gegeben. Von diesen müssen von 3er-Gruppen eins und von 4er-Gruppen zwei umgesetzt werden. Alle weiteren Zusatzfeatures gelten als Wahlfeatures und sind 5 Punkte wert.

Bewertungskriterien

Folgende Kriterien fließen in die Bewertung ein. Die Endnote bestimmt sich durch die Gesamtzahl der Punkte, die erreicht wurde. Für ein Bestehen ist es aber zudem Voraussetzung, dass in den Kategorien 1.,2.,3. jeweils mindestens die Hälfte der Maximalpunktzahl erreicht wurde.

1. **Produktbezogene Qualitätsaspekte (50 P.)**
   1. Funktionalität und Zuverlässigkeit (30 P.)
      1. Implementierung aller Pflichtfeatures (14 P.)
      2. Implementierung ausgewählter Wahlfeatures (6 P.)
      3. Fehlerfreiheit der implementierten Funktionen (10 P.)
   2. Gebrauchstauglichkeit (15 P.)
      1. Verständlichkeit, Erlernbarkeit, Bedienbarkeit und Attraktivität des Produkts
      2. Abfangen von Bedienfehlern
   3. Effizienz/Performanz (5 P.)
      1. Keine unverhältnismäßig lange Rechen-/Antwortzeit
      2. Angemessener Ressourcenverbrauch (Arbeitsspeicher etc.)
2. **Entwicklungsbezogene Qualitätsaspekte (30 P.)**
   1. Stabile und modifizierbare Architektur (10 P.)
   2. Testing (8 P.)
      1. Testqualität
      2. Prozentuale Testabdeckung
   3. Einhaltung Metriken (4 P.)
   4. Codingstyle (4 P.)
      1. Einhaltung Codierungskonventionen
      2. Verständlicher Code (sprechende Variablen-/Methodennamen, Kommentare wo nötig)
      3. Einfacher Code (wo möglich)
   5. Versionsverwaltung (4 P.)
      1. Sprechende Commit-Nachrichten
      2. Sinnvolle Commitgrößen
3. **Dokumentation (20 P.)**
   1. Lückenlose Dokumentation der Anforderungen (Usecase-Diagramm, Storycards) (3 P.)
   2. Qualitativ hochwertige Dokumentation der Architektur inkl. Begründung (7 P.)
   3. Bedienungsanleitung für das Produkt (2 P.)
   4. Javadoc-Dokumentation (4 P.)
   5. Abschlusspräsentation (4 P.)

Nicht-funktionale Anforderungen (jedes Release)

Codequalität

* Einhaltung der Metriken wie im Abschnitt "Codingstyle, Metriken, Testabdeckung" beschrieben
* Einhaltung Style-Convention wie im Abschnitt "Codingstyle, Metriken, Testabdeckung" beschrieben
* Kommentierung des Codes, wo notwendig

Testabdeckung

* Abdeckung von 80 % des Codes (ohne GUI Klassen) durch JUnit-Tests

Dokumentation

* Stets aktualisierte Anforderungsdokumentation via Story-Cards, Anwendungsfalldiagrammen
* Stets aktualisierte Dokumentation der Architektur unter Zuhilfenahme von UML-Diagrammen (Klassen-, Objekt-, Sequenz-, Zustand-), wo sinnvoll.
* Stets aktualisierte Javadoc-Dokumentation des Programms
* Stets aktualisierte Dokumentation der Programmverwendung (Bedienungsanleitung)

Beteiligung in Moodleoverflows

Die Stackoverflow-artigen Moodle-Foren (Moodleoverflows) dienen der gegenseitigen Hilfe der Studierenden und sorgen für effektivere Softwareentwicklung über alle Gruppen hinweg. Durch Beteiligung an den Moodleoverflows können bis zu 5 Bonuspunkte erreicht werden. Diese Punkte werden am Ende des Praktikums anhand der Aktivität der Nutzer in den Moodleoverflows vergeben.

Dokumentation

Die Doku ist kein Wahlfeature!

Anforderungsdokumentation

Die Anforderungen des Produkts sollen übersichtlich in einem Anwendungsfalldiagramm dargestellt werden. Die einzelnen Anforderungen sollen detailliert auf Story-Cards festgehalten werden, auf die während des Praktikums zurückgegriffen werden soll.

Bedienungsanleitung

In diesem Dokument sollen alle Funktionen des Systems kurz vorgestellt werden und erklärt sein, wie diese angesteuert werden können.

Generell gilt: Sind Features nicht korrekt in der Bedienungsanleitung beschrieben und werden Sie bei der Bewertung nicht entdeckt, werden sie als nicht vorhanden bewertet. Analoges gilt, wenn sich die Bedienung der Features nicht unmissverständlich aus der Bedienungsanleitung ergibt.

Architekturdokumentation

In diesem Dokument sollen die grundsätzlichen Ideen hinter der gewählten Architektur dargelegt werden. Es sollte erklärt werden, in welche Komponenten das System eingeteilt wurde, welche Klassen zur Umsetzung der einzelnen Komponenten dienen und welche Kommunikation zwischen den einzelnen Klassen stattfindet. Außerdem sollte klar werden, in wie fern die gewählte Architektur eine Erweiterbarkeit und einfache Wartbarkeit des Programms gewährleistet.

Für die Dokumentation kann die Verwendung von UML-Diagrammen sinnvoll sein.

Javadoc-Dokumentation

Alle öffentlichen Klassen, Methoden (abgesehen von Gettern und Settern) und Attribute sollten Javadoc-konform dokumentiert sein.

Codingstyle, Metriken, Testabdeckung

Das Template verwendet die Plugins [PMD](https://github.com/jqno/jacoco-badge-plugin), [JaCoCo](https://www.eclemma.org/jacoco/)

Wählbares Terror Theme (3P.)

Es gibt wenigstens 2 Unterschiedliche grafische Themen (z.B. Untote, alles Blutig, etc.). Zwischen diesen Themen lässt sich vor und während des Spiels das Aussehen des Spiels verändern (z.B. über das Menü).

Anzahl der Bausteine wählbar (2P.)

Vor dem Spiel kann ausgewählt werden, wie viele Bausteine (Kuppel und Level 1-3) es gibt. Das Spiel kann demnach früher/später dadurch enden, dass ein Spieler nicht mehr Bauen kann.

Kompetente KI (3P.)

Bei der Wahl eines Computerspielers kann man dessen Spielstärke einstellen. Ein starker Computerspieler sollte versuchen möglichst schnell zu gewinnen, wenn er der Meinung ist, vor allen anderen Mitspielern gewinnen zu können, und ansonsten versuchen die anderen Spieler zu behindern.

Anzahl der Arbeiter wählbar (3P.)

Vor Spielstart kann ausgewählt werden, wie viele Arbeiter jeder Spieler zur Verfügung hat.

Größe des Spielfeldes Wählbar (2P.)

Die Größe des Spielfeldes kann vor Spielbeginn festgelegt werden.

Die Welt ist eine "Kugel" (3P.)

Arbeiter können das Spielfeld am Rand verlassen und dafür das entsprechende Feld am gegenüberliegenden Rand betreten.

Macht der Götter Simple (1P. je 2 Götterkarten, Max 5 Punkte)

Die Spielvariante "Macht der Götter" kann den Regeln entsprechend gewählt werden. Je 2 vollständig und richtig implementierter "Simple Gods" Götter gibt es einen Punkt.

Macht der Götter Advanced (1P. je 4 Götterkarten, Max 5 Punkte)

Die Spielvariante "Macht der Götter" kann den Regeln entsprechend gewählt werden. Je 4 vollständig und richtig implementierter "Advanced Gods" Götter gibt es einen Punkt.

Das Template verwendet die Plugins [PMD](https://pmd.github.io/latest/pmd_userdocs_tools_maven.html) und [JaCoCo](https://www.eclemma.org/jacoco/) zur Generierung von Reports über Metriken, Codestyle und Testabdeckung. Die Verwendung der Plugins und der resultierenden Reports wird im Tutorial demonstriert.

Es wird eine Testabdeckung des gesamten Codes von 90% Instruction Coverage erwartet. (Dabei dürfen GUI-Klassen ausgeschlossen werden. Weitere Details dazu finden sich im Tutorial.)

Im Template wird [unser PMD-Regelsatz](https://projects.isp.uni-luebeck.de/software-engineering-praktikum-dokumente/pmd) eingebunden. Diese Regeln sind für den Code und die Testfälle einzuhalten.