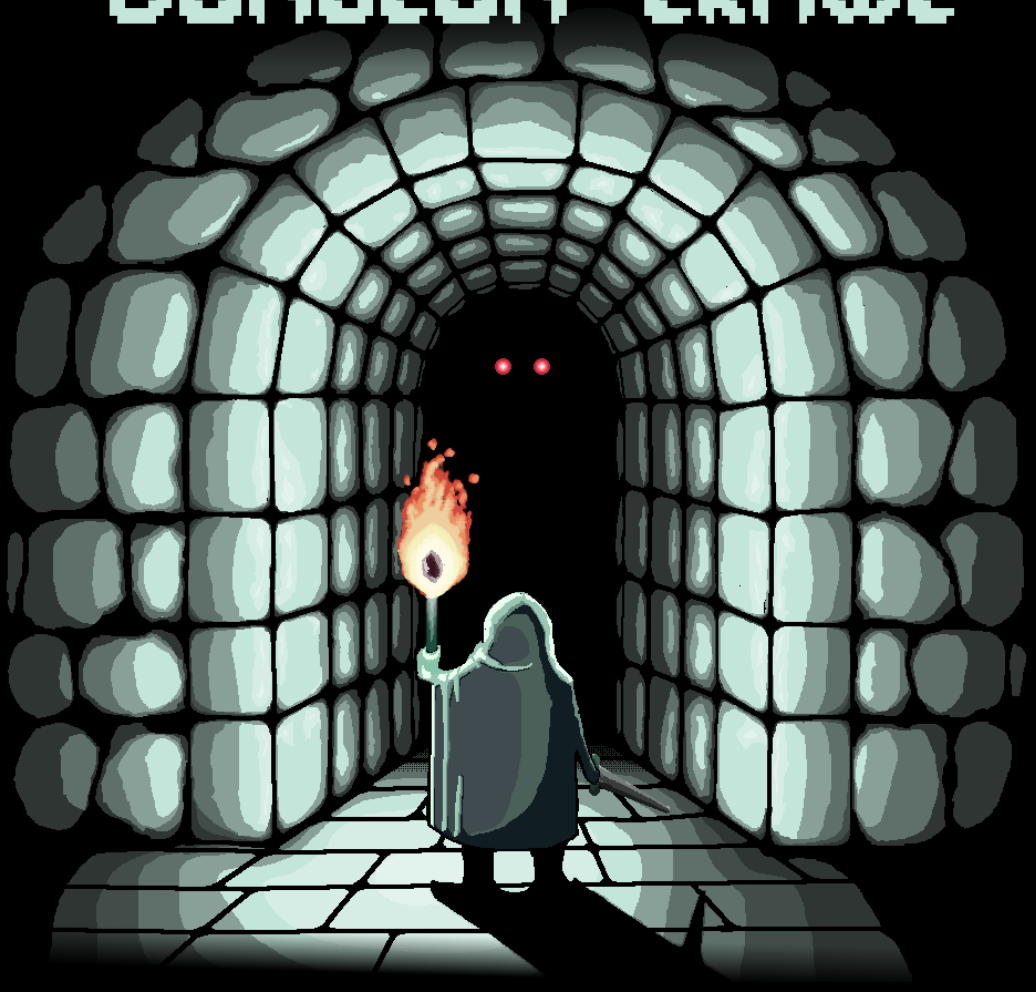
ZHAW School of Engineering

Bachelor of Science in Informatik

Software-Projekt PM4 FS25

Projektskizze

DUNGEON CRAWL



16.03.2025

Jakub Baranec

Anujan Chandrawathanan

Trong-Nghia Dao

Nino Frei

Nicola Meier

Dominic Odermatt

Lucien Perret

Tobias Uetz

Jil Zerndt

Tobias Zülli

Inhaltsverzeichnis

1 Projektidee

In diesem Abschnitt wird die Projektidee zusammen mit dem Kontext der Ausgangslage erläutert. Ausserdem wird auf den Kundennutzen, den Stand der Technik und die Konkurrenz eingegangen.

1.1 Ausgangslage

Der Videospielmarkt hat seit seinen Anfängen in den 1970er Jahren ein stetiges Wachstum erfahren und sich weltweit etabliert. Bereits in den frühen Jahren waren Arcade-Spiele wie *Pong* und *Space Invaders* prägend für die Industrie und legten den Grundstein für die heutige Vielfalt an Spielen. [1]

In den 2000er Jahren dominierten sogenannte “AAA” (Triple-A) Entwicklerstudios den globalen Markt. Diese Studios produzierten Spiele mit Produktionskosten, die bis zu 40 Millionen Dollar erreichen konnten. Ein Beispiel hierfür ist *Halo 3*, dessen Entwicklungskosten auf 30 Millionen Dollar geschätzt wurden, ergänzt durch ein Marketingbudget von 40 Millionen Dollar [2].

Ab den 2010er Jahren stieg die Popularität von unabhängigen Entwicklern (Indie-Devs) deutlich an. Diese Entwickler, oft kleine Teams mit begrenzten Budgets, profitierten von digitalen Vertriebsplattformen wie Steam, die es ihnen ermöglichten, ihre Spiele ohne traditionelle Publisher direkt an die Spieler zu verkaufen. Der Erfolg von Indie-Spielen wie *Super Meat Boy* und *Braid* verdeutlicht diesen Trend. [3]

Indie-Games sind in der Regel zu einem günstigeren Preis erhältlich als AAA-Titel und bieten dennoch oft viele Stunden Spielspass. Viele Spieler bevorzugen Indie-Spiele aufgrund ihres niedrigeren Preises und des persönlicheren Charakters, den sie vermitteln. [4]

1.2 Idee

Dungeon Crawl soll ein Rogue-like Indie-Game werden, in dem sich der Spieler durch ein Labyrinth voller Gefahren kämpft. Das Hauptziel besteht darin, einen möglichst hohen High-Score aufzustellen, indem der Spieler Gegner besiegt, Geheimnisse entdeckt und sich so lange wie möglich am Leben hält.

Durch die prozedurale Generierung des Labyrinths wird jede Runde einzigartig sein, sodass kein Durchlauf dem anderen gleicht. Dies sorgt für eine hohe Wiederspielbarkeit und fordert den Spieler immer wieder aufs Neue heraus. Zufällige Raumlayouts, unterschiedliche Feindplatzierungen und dynamische Events garantieren eine spannende und unvorhersehbare Erfahrung.

Der Kampf im Labyrinth gegen lauernde Monster findet in einem rundenbasierten Kampfsystem statt, dass Strategie und stetige Charakteroptimierung fordern. Jeder Angriff, jede Fähigkeit und jede Bewegung müssen gut durchdacht sein, um möglichst effizient zu kämpfen und Ressourcen zu schonen. Unterschiedliche Gegnertypen mit einzigartigen Verhaltensweisen sorgen für taktische Herausforderungen, die von den Spielern kreative Lösungen und Anpassungen erfordern.

Nach dem Tod des Spielercharakters wird der erzielte Score in Verbesserungspunkte umgewandelt. Diese Verbesserungspunkte ermöglichen es dem Spieler, permanente Verbesserungen freizuschalten, die zukünftige Durchläufe erleichtern oder neue Spielstile ermöglichen. Dazu gehören beispielsweise verstärkte Charaktereigenschaften, neue Fähigkeiten oder alternative Charakterklassen.

Um eine herausfordernde, aber belohnende Spielerfahrung zu schaffen, soll das Gameplay durch eine Mischung aus strategischem Kampf, Ressourcenmanagement und geschicktem Navigieren im Labyrinth geprägt sein. Jeder Fehler ist eine Lektion, und jeder Fortschritt fühlt sich bedeutungsvoll an.

1.3 Kundennutzen

Dungeon Crawl richtet sich an Spieler, die herausfordernde und abwechslungsreiche Spiele mit hoher Wiederspielbarkeit schätzen. Jede Runde wird eine andere Erfahrung als die letzte, jedes Labyrinth wird einzigartig sein. Dies spricht besonders Fans von Roguelikes, und rundenbasierten taktischen Kämpfen an.

Durch das Verbesserungspunkte-System entsteht eine langfristige Motivation, da Spieler auch nach dem Scheitern Fortschritte machen und sich für zukünftige Runden verbessern können. Dies sorgt für eine sinnvolle Belohnungsstruktur und eine stetige Weiterentwicklung der Spielfigur, was Frustration verringert, und den Spieler ermutigt sich immer wieder neuen Herausforderungen zu stellen.

1.4 Stand der Technik / Konkurrenzanalyse

Der Markt für Rogue-likes hat in den letzten Jahren stark an Popularität gewonnen. Erfolgreiche Titel wie *The Binding of Isaac*, *Slay the Spire* oder *Darkest Dungeon* haben gezeigt, dass prozedural generierte Herausforderungen, taktische Kämpfe und langfristige Progressionssysteme eine grosse Fangemeinde ansprechen. Diese Spiele setzen auf hohe Wiederspielbarkeit und kombinieren strategische Kämpfe mit einer stetigen Charakterentwicklung.

Im Vergleich zu bestehenden Titeln soll *Dungeon Crawl* mit einem tiefgehenden rundenbasierten Kampfsystem, das sowohl taktische Tiefe als auch zugängliche Mechaniken bietet, eine eigene Nische besetzen. Zudem wird ein Motivationssystem durch permanente Upgrades integriert, um den Spielfortschritt zu belohnen und Frustration zu minimieren. Die Mischung aus strategischem Gameplay, zufälligen Herausforderungen und einer langfristigen Progression bietet eine spannende Alternative zu klassischen Rogue-likes und sorgt für eine motivierende Spielerfahrung.

2 Kontextszenario (Hauptablauf)

Leon Kämpfer kommt nach einem langen Tag nach Hause. Er legt seine Tasche beiseite, macht es sich bequem und startet seinen Gaming-PC. Heute ist ein besonderer Tag, denn er hat sich das neu erschienene Spiel *Dungeon Crawl* gekauft – ein düsteres, Rogue-like Dungeon-Abenteuer, das ihn sofort in seinen Bann zieht.

Gespannt startet er das Spiel. Im Hauptmenü wählt er “Neues Spiel” aus und ein kurzes textbasiertes Intro setzt die Szene: Ein namenloser Held, der Unknown Hero, erwacht in den Tiefen eines mysteriösen Dungeons. Ohne Erinnerung an seine Vergangenheit muss er sich den Gefahren der finsteren Gewölbe stellen. Die Reise beginnt.

Leon navigiert den Unknown Hero vorsichtig durch die erste Ebene des Dungeons. Die Karte deckt sich nach und nach auf, während er sich durch die dunklen Korridore bewegt. Plötzlich taucht ein Skelett auf – ein niederer Untoter, aber dennoch eine Bedrohung. Der Kampf beginnt.

Der Unknown Hero greift mit seinem Schwert an – ein präziser Hieb trifft das Skelett und verursacht 5 HP-Schaden. Doch der Untote gibt nicht auf und schlägt zurück. Geistesgegenwärtig blockt der Unknown Hero den Angriff, reduziert den Schaden und verliert nur 2 HP. In der zweiten Runde setzt er erneut zum Angriff an: Ein weiterer Schwerthieb, und das Skelett zerfällt in seine Einzelteile.

Leon lootet den besiegten Gegner. Neben ein paar Goldmünzen findet er ein Paar abgetragene Stiefel. Diese bieten +1 auf Rüstung – besser als seine aktuellen. Er öffnet das Inventar und rüstet seinen Helden mit den neuen Stiefeln aus. Gut gerüstet setzt er seine Erkundung fort und erreicht schliesslich den Eingang zur zweiten Ebene des Dungeons.

Die Herausforderungen werden schwieriger. Leon navigiert den Unknown Hero tiefer in die Dunkelheit, bis er auf einen Draugr trifft – ein weitaus gefährlicherer Gegner als das zuvor besiegte Skelett. Der Kampf beginnt, doch der Draugr erweist sich als zäher Widersacher. Die Auseinandersetzung zieht sich in die Länge, und schliesslich erliegt der Unknown Hero seinen Wunden. Mit einem letzten Hieb fällt er, und der Draugr beendet das Gefecht. Game Over.

Doch das Abenteuer ist nicht vergebens. Durch seine bisherigen Erfolge hat Leon Verbesserungspunkte erhalten, die er im Verbesserungsbaum ausgeben kann. Mit einem Punkt verstärkt er die HP seines Helden – eine kleine, aber wertvolle Verbesserung für den nächsten Versuch. Er startet einen neuen Lauf, entschlossen, weiterzukommen.

Nach mehreren Versuchen und zwei Stunden Spielzeit beendet Leon das Spiel. Trotz des schwierigen Verlaufs fühlt er sich erfüllt – er hat Fortschritte gemacht, seinen Charakter verbessert und sich selbst herausgefordert. Zufrieden lehnt er sich zurück, bereits voller Vorfreude auf das nächste Abenteuer.

3 Weitere Anforderungen

Dieser Abschnitt erläutert die funktionalen und nicht-funktionalen Anforderungen, die für das Spiel erforderlich sind. Funktionale Anforderungen definieren spezifische Funktionen und Fähigkeiten, während nicht-funktionale Anforderungen FURPS und andere Qualitätsmerkmale betreffen.

3.1 Funktionale Anforderungen

Die funktionalen Anforderungen definieren die zentralen Mechaniken und Interaktionen, die das Spielerlebnis prägen. Jede Anforderung stellt sicher, dass die Spieler auf strukturierte und ansprechende Weise durch die Level vorankommen können.

3.1.1 Prozedural generierte Karten

Beim Start eines neuen Spiels oder Levels generiert das System eine zufällige Dungeon-Karte mit Gängen und Gegnern. Diese sind zunächst zugedeckt, werden aber allmählich durch den Spieler aufgedeckt.

3.1.2 Verbesserungsbaum

Der Verbesserungsbaum erlaubt es dem Spieler Verbesserungspunkte zu investieren, um Charakter-Verbesserungen zu erhalten. Diese können seine grundlegenden Statuswerte erhöhen oder zusätzliche Fähigkeiten freischalten.

3.1.3 Inventarsystem

Der Spieler kann Gegenstände sammeln, ausrüsten und aus dem Inventar verwenden, um seinen Charakter zu verbessern oder einen speziellen Effekt auszulösen. Diese Gegenstände werden in einem Inventarsystem verwaltet, das dem Spieler Zugriff auf diese gewährt.

3.1.4 Rundenbasiertes Kampfsystem

Der Spieler kann einen Kampf auslösen, in dem er auf einen Gegner zuläuft. Dies startet eine Instanz des rundenbasierten Kampfsystems, in welchem der Spieler und der Gegner abwechselnd eine Aktion ausführen können.

3.2 Nicht-Funktionale Anforderungen

Nicht-funktionale Anforderungen konzentrieren sich eher auf die Leistung, die Benutzerfreundlichkeit und das Gesamterlebnis des Spiels, statt spezifischen Funktionen. Diese stellen sicher, dass das Spiel als Gesamtes qualitativ hochwertig ist und ein zufriedenstellendes Spielerlebnis bietet.

3.2.1 Funktionalität

Die Spielmechaniken müssen korrekt funktionieren und logisch miteinander interagieren.

3.2.2 Benutzerfreundlichkeit

Das Spiel muss sicherstellen, dass die Steuerung sowie das Menü des Spiels intuitiv, klar und übersichtlich sind. Zudem wird eine Spielanleitung beigelegt, welche alle Mechaniken des Spiels genau erklärt.

3.2.3 Verlässlichkeit

Das Spiel darf nicht abstürzen und der Spieler darf keinen Spielfortschritt verlieren. Dies soll auch gewährleistet sein, wenn der Spieler das Spiel abrupt beendet.

3.2.4 Leistung

Das Spiel muss flüssig und ohne erkennbaren Lag funktionieren. Dies muss auch gewährleistet sein, wenn das Spiel auf grosse Levels und mehrere Gegner skaliert wird.

3.2.5 Unterstützbarkeit

Der Code sollte modular aufgebaut sein, sodass neue Inhalte oder Bugfixes leicht implementiert werden können. Ebenfalls sollte der Code und die Entscheidungen während des Entwicklungsprozesses ausreichend dokumentiert sein.

3.2.6 Skalierbarkeit

Das Spiel sollte problemlos um neue Karten, Gegner oder Mechaniken erweitert werden können, ohne das bestehende System zu beeinträchtigen.

3.2.7 Wiederspielwert

Durch zufällig generierte Level, verschiedene Fähigkeiten, Gegenstände und Gegner soll das Spiel langfristig unterhaltsam bleiben.

4 Projektmanagement

Hier werden die für das Projektmanagement relevanten Inhalte erläutert, darunter die verfügbaren Ressourcen, die Grobplanung, bestehende Risiken sowie die potenzielle Wirtschaftlichkeit. Dies dient dazu, organisatorische Unklarheiten zu klären und das Projekt auf einem hohen qualitativen Standard zu halten.

4.1 Ressourcen

Wir planen, unser Code-Repository auf GitHub zu hosten und für CI/CD GitHub Actions zu nutzen. Zusätzlich setzen wir Sonarqube ein, um eine hohe Codequalität und Clean Code zu gewährleisten. Mit Meson testen wir unseren C-Code und stellen eine hohe Testabdeckung sicher.

Zudem stehen uns Kubernetes und ArgoCD zur Verfügung, falls wir weitere Tools für Deployment oder andere Anforderungen benötigen.

Unser Team verfügt über grundlegende Kenntnisse in C, wobei einige Mitglieder bereits Grundlagenwissen in relevanten Tools oder vertiefte Erfahrung mit C gesammelt haben. Wenig bis keine Erfahrung hat das Team bei Sonarqube und die CI/CD mittels GitHub Actions.

Die Entscheidung fiel auf C, da wir bewusst auf Java, Python und JavaScript verzichten wollten und das Team mindestens über grundlegende Kenntnisse in C verfügt.

4.2 Roadmap

Da unser Team nach den Prinzipien von Agile Software Development und SCRUM arbeitet, werden wir keinen genauen Zeitplan erstellen. Stattdessen haben wir eine Roadmap erstellt in der Meilensteine und deren Kriterien aufgeführt sind. Unsere Teammitglieder werden nicht in fixen Gruppen arbeiten, sondern dynamisch den anstehenden Aufgaben zugeteilt.

Datum	Milestone	Kriterien
30.03.	Erste funktionale Struktur	Jede für das Spiel notwendige Grundfeature wurde umgesetzt. • Der Spieler kann sich durch ein prozedural generiertes Labyrinth bewegen. • Der Spieler kann einen Kampf mit einem Gegner starten und ihn angreifen. • Der Spieler kann durch eine Tür laufen und in ein neues Labyrinth gelangen.
27.04.	Minimum Viable Product	Die Grundfeatures des Spiels wurden zu einer zufriedenstellenden Spielerfahrung ausgebaut. • Die Labyrinthe sind abwechslungsreich und es sind verschiedene Gegnertypen implementiert. • Das Kampfsystem ermöglicht Strategie und Taktik durch eine Vielfalt an Möglichkeiten. • Jedes Level ist schwieriger als das Letzte. • Alle Unit Tests sind erfolgreich und unsere Testabdeckung ist über 90%.
11.05.	Funktionale Erweiterungen	Ein Verbesserungssystem ist hinzugefügt worden, welches das Freischalten von permanenten Verbesserungen ermöglicht.

		Ausserdem wurden die Grundfeatures mit zusätzlicher Funktionalität erweitert.
23.05.	Qualitative Verbesserungen / Projektabgabe	• Das Spiel wurde zu einem ansprechenden Endprodukt, welches Spass macht, vervollständigt. • Alle Unit Tests sind erfolgreich und unsere Testabdeckung ist über 90%.

Tabelle 1: Roadmap / Grobplanung

4.3 Risiken

Das Projekt ist mit mehreren Risiken konfrontiert. Deren Auswirkungen müssen während der Entwicklung laufend berücksichtigt und überwacht werden. Es handelt sich dabei um organisatorische sowie technische Risikofaktoren. Die folgende Tabelle gibt einen Überblick über alle identifizierten Risiken inklusive deren Eintrittswahrscheinlichkeit und Schweregrad.

Risiko	Beschreibung	Eintritt	Auswirkung
Schlechte Koordination im Team	Das Team hat 10 Mitglieder und ist daher anfällig für ineffiziente Kommunikation und fehlende Koordination.	Mittel	Mittel
Technische Probleme bei Entwicklung durch fehlende Erfahrung	Nicht alle Teammitglieder verfügen über die gleiche Erfahrung mit den eingesetzten Technologien. Für weniger erfahrene C Entwickler bietet das Projekt eine hohe technische Komplexität, was die Entwicklung und Fehlerbehebung anspruchsvoll macht.	Mittel	Mittel
Fehlende Balance im Gameplay	Da das Spiel auf prozeduraler Generierung, zufälligen Events und komplexen Spielmechaniken basiert, besteht die Gefahr, dass das Gameplay für den Spieler unfair wird.	Hoch	Tief
Fehlende Integration und Verfügbarkeit der Infrastruktur	Eine fehlerhafte Konfiguration oder mangelnde Verfügbarkeit der verschiedenen Tools und Infrastrukturkomponenten könnte zu Verzögerungen, fehlerhaften Builds, ineffizientem Deployment und einer insgesamt schlechteren Entwicklungs- und Testumgebung führen.	Tief	Hoch

Tabelle 2: Überblick Risiken

4.4 Wirtschaftlichkeit

Das Projekt ist kein Leistungsauftrag eines Praxispartners und daher nicht vergütet. Die Aufwände werden in Stunden erfasst. Der geplante Aufwand für das gesamte Projekt beträgt 1’200 Stunden gesamthaft, oder 120 Stunden pro Teammitglied. Eine Planung des Aufwandes wird in Kapitel 4.2 beschrieben. Allfällige wirtschaftliche Erträge können erst in der nächsten Phase geschätzt werden. Möglichkeiten der nächsten Phase werden in Kapitel 5 beschrieben.

Das Projekt profitiert von der Nutzung moderner Technologien wie GitHub Actions für die CI/CD-Pipeline, Meson für die Tests und SonarQube zur Sicherstellung der Codequalität. Diese

Technologien ermöglichen eine hohe Automatisierung der Build- und Testprozesse. Durch den Einsatz von SCRUM kann das Team in klar definierten Sprints arbeiten, was eine schnelle Anpassung an neue Anforderungen und kontinuierliche Fortschritte ermöglicht. Die agile Methodik sorgt dafür, dass das Team flexibel und fokussiert bleibt, wobei regelmässig überprüft wird, ob die entwickelten Features den Anforderungen entsprechen. Die Verwendung von Kubernetes und ArgoCD für das Deployment und die Skalierung ist optional, könnte aber zusätzlich dazu beitragen, die Infrastruktur langfristig effizient und skalierbar zu gestalten.

Insgesamt wird durch den gezielten Einsatz dieser Technologien die Produktivität gesteigert, die Codequalität gesichert und die Projektzeit effizient genutzt, sodass das Projekt innerhalb der 1’200 Stunden wirtschaftlich realisiert werden kann.

5 Ausblick

Während die Entwicklung von Dungeon Crawl voranschreitet, stehen verschiedene Möglichkeiten für die nächste Phase im Raum. Ein öffentlicher Playtest könnte eine wertvolle Gelegenheit sein, frühzeitig Feedback zu sammeln und das Spiel gemeinsam mit unserer Community zu verfeinern. Dadurch liessen sich Balancing-Anpassungen vornehmen, Bugs frühzeitig entdecken und das Spielerlebnis weiter optimieren.

Langfristig wäre auch ein Release auf einer Plattform wie Steam denkbar, um das Spiel einem grösseren Publikum zugänglich zu machen. Ob dieser Schritt als Early-Access-Version oder als vollständiger Release erfolgt, hängt stark von den Ergebnissen des Playtests und der weiteren Entwicklung ab. Unser Ziel bleibt es, das bestmögliche Spielerlebnis zu bieten – mit einer soliden Grundlage, die durch das Feedback der Spieler stetig wächst und verbessert wird.

Zum Abschluss von PM4 wird das Spiel spielbar sein und durch eine Bedienungsanleitung ergänzt, die die wichtigsten Funktionen erklärt und den Einstieg erleichtert. Dieses NoTech-Produkt unterstützt den Praxispartner bei der Nutzung und Präsentation des Spiels.

6 Quellenverzeichnis

[1] S. L. Kent, The ultimate history of video games : from Pong to Pokémon and beyond : the story behind the craze that touched our lives and changed the world. Roseville, Calif. : Prima Pub., 2001. Zugegriffen: 16. März 2025. [Online]. Verfügbar unter: http://archive.org/details/ultimatehistoryo0000kent

[2] J. Schreier, Blood, sweat, and pixels : the triumphant, turbulent stories behind how video games are made. New York : HarperCollins Publishers, 2017. Zugegriffen: 16. März 2025. [Online]. Verfügbar unter: http://archive.org/details/bloodsweatpixels0000schr

[3] J. Juul, Handmade Pixels: Independent Video Games and the Quest for Authenticity. Cambridge, MA: The MIT Press, 2019.

[4] N. Lipkin, «Examining Indie's Independence: The Meaning of ‹Indie› Games, the Politics of Production, and Mainstream Cooptation», Loading..., Bd. 7, Nr. 11, Art. Nr. 11, 2013, Zugegriffen: 16. März 2025. [Online]. Verfügbar unter: https://journals.sfu.ca/loading/index.php/loading/article/view/122

7 Abbildungs- und Tabellenverzeichnisse

7.1 Abbildungsverzeichnis

7.2 Tabellenverzeichnis

8 Glossar

A

- **ArgoCD**
  - Ein Kubernetes-natives Tool zur kontinuierlichen Bereitstellung (Continuous Delivery) von Anwendungen.
- **AAA-Spiele (Triple-A)**
  - Hochbudgetierte Videospiele grosser Entwicklerstudios mit aufwendiger Produktion und grossem Marketingaufwand.
- **Arcade-Spiele**
  - Frühe Videospiele, die in Spielhallen auf Münzautomaten spielbar waren, z.B. *Pong* und *Space Invaders*.

B

- **Balancing-Anpassung**
  - Veränderungen an Spielmechaniken, um das Spiel fairer oder herausfordernder zu gestalten.
- **Benutzerfreundlichkeit (Usability)**
  - Das Mass, in dem eine Anwendung oder ein System leicht und intuitiv genutzt werden kann.
- **Bugs**
  - Programmfehler oder unerwartete Fehlfunktionen in einer Software.

C

- **CI/CD (Continuous Integration / Continuous Deployment)**
  - Ein Entwicklungsprozess, bei dem Codeänderungen kontinuierlich integriert, getestet und automatisch bereitgestellt werden.
- **Community**
  - Die Gemeinschaft von Spielern, Entwicklern und Fans, die ein Spiel unterstützen und mitgestalten.
- **Clean Code**
  - Ein Programmierstil, der sich durch einfache, verständliche und wartbare Code-Strukturen auszeichnet.

D

- **Deployment**
  - Der Prozess, eine Softwareanwendung bereitzustellen und auszuführen.
- **Digitale Vertriebsplattform**
  - Eine Online-Plattform, die den Kauf und Download von Spielen ermöglicht, z.B. Steam, Epic Games Store oder GOG.
- **Draugr**
  - Ein untoter Gegner in Dungeon Crawl, inspiriert von nordischer Mythologie.

E

- **Early-Access-Version**
  - Eine unfertige, aber spielbare Version eines Spiels, die frühzeitig veröffentlicht wird, um Feedback von Spielern zu sammeln.

F

- **FURPS**

- FURPS dient der Kategorisierung von Softwareanforderungen und umfasst Funktionalität, Benutzerfreundlichkeit, Zuverlässigkeit, Leistung und Wartbarkeit.

G

- **Gameplay-Balance**
  - Die Abstimmung von Spielmechaniken, um eine faire und unterhaltsame Spielerfahrung zu gewährleisten.
- **GitHub**
  - Eine Plattform zur Versionsverwaltung und Zusammenarbeit an Softwareprojekten, die auf Git basiert.
- **GitHub Actions**
  - Ein Automatisierungswerkzeug innerhalb von GitHub, das CI/CD-Prozesse ermöglicht.

H

- **High-Score**
  - Eine Punktzahl, die durch das Erreichen bestimmter Ziele im Spiel erzielt wird und als Massstab für den Erfolg dient.

I

- **Indie-Game**
  - Ein Videospiel, das von unabhängigen Entwicklern oder kleinen Studios ohne Unterstützung eines grossen Publishers entwickelt wurde.
- **Infrastructure as Code (IaC)**
  - Ein Ansatz zur Verwaltung von IT-Infrastrukturen über maschinell lesbare Konfigurationsdateien anstelle manueller Prozesse.

J

K

- **Kubernetes**
  - Ein Open-Source-System zur Automatisierung der Bereitstellung, Skalierung und Verwaltung von containerisierten Anwendungen.

L

- **Lag**
  - Eine Verzögerung in der Reaktionszeit eines Spiels, oft verursacht durch Netzwerkprobleme oder schlechte Hardwareleistung.

M

- **Meson**
  - Ein Build-System zur effizienten Kompilierung und Verwaltung von Softwareprojekten, insbesondere für C und C++.
- **Minimum Viable Product (MVP)**
  - Eine frühe Version eines Produkts mit minimalen aber funktionsfähigen Features, die für erste Tests und Feedback genutzt werden kann.

N

O

P

- **Playtest**

    - Ein Testlauf eines Spiels durch Spieler, um Feedback zu sammeln und Probleme zu identifizieren.
- **Prozedurale Generierung**
    - Ein Verfahren zur automatischen Erstellung von Inhalten, z.B. in Videospielen, durch Algorithmen statt durch manuelles Design.
- **Projektmanagement**
    - Der Prozess der Planung, Organisation und Überwachung eines Projekts, um definierte Ziele innerhalb eines bestimmten Zeitraums und Budgets zu erreichen.
- **Progressionssystem**
    - Ein Mechanismus, der es Spielern ermöglicht, Fortschritte zu machen, z.B. durch Erfahrungspunkte, neue Fähigkeiten oder bessere Ausrüstung.

Q

R

- **Risikomanagement**
    - Der Prozess der Identifikation, Analyse und Bewältigung potenzieller Probleme innerhalb eines Projektes.
- **Rogue-like**
    - Ein Videospielgenre mit prozedural generierten Levels, Permadeath (dauerhafter Tod des Charakters) und rundenbasierten Kämpfen.
- **Release**
    - Die Veröffentlichung eines Spiels oder Softwareprodukts für die Öffentlichkeit.
- **Ressourcen**
    - Alle verfügbaren Mittel, die für die Durchführung eines Projekts genutzt werden können, z.B. Software, Hardware, Zeit und Personal.

S

- **SCRUM**
    - Ein agiles Framework zur Softwareentwicklung, das iterative Entwicklungszyklen (Sprints) nutzt, um schnelle Anpassungen zu ermöglichen.
- **Steam**
    - Eine digitale Vertriebsplattform für PC-Spiele, die den Kauf, Download und die Verwaltung von Spielen ermöglicht.
- **Skill**
    - Eine Fähigkeit oder Fertigkeit, die ein Spielercharakter im Spiel erlernen oder verbessern kann.
- **Skilltree**
    - Ein Fortschrittssystem, das Spielern ermöglicht, Fähigkeiten in einem verzweigten System freizuschalten.
- **SonarQube**
    - Ein Tool zur Analyse von Quellcode, das Codequalität und Sicherheitsprobleme erkennt.
- **Sprint**
    - Ein festgelegter Zeitraum in der agilen Softwareentwicklung, in dem ein definiertes Arbeitsziel erreicht werden soll.

T

- **Testabdeckung**
  - Ein Mass dafür, wie viele Teile eines Programmcodes durch automatisierte Tests überprüft werden.

U

- **Unit Testing**
  - Ein Softwaretest, mit dem einzelne, abgrenzbare Teile von Computerprogrammen überprüft werden.

V

- **Verbesserungspunkte**
  - Punkte, die nach einem Spieldurchlauf erhalten werden und für dauerhafte Upgrades oder Verbesserungen genutzt werden können.

W

- **Wirtschaftlichkeit**
  - Die Bewertung, ob ein Projekt oder eine Investition in Bezug auf den eingesetzten Aufwand und den erwarteten Nutzen lohnenswert ist.

X

Y

Z