Dokumentation M431

05.07.2024

GAME ERSTELLEN MIT GODOT

von Dario Kulici & Martin Sorrenti

DOKUMENTATION PROJEKT VIDEOSPIEL

EINLEITUNG

Wir machen ein Videospiel. Kein einfaches Spiel auf Scratch, sondern ein richtiges Videospiel auf einer Game Engine. Eine Game Engine bietet Fortgeschrittene Methoden und Features an, die wir nutzen können, um unser Spiel nach Belieben zu erstellen. Wir verfolgen das Ziel, ein fertiges, bugfreier,

unterhaltsames 2D-Videospiel zu erstellen. Wir arbeiten zusammen und teilen die Aufgaben untereinander auf, um so effizient wie möglich voranzukommen.

UNSER ENTWICKLERTEAM

DARIO

Ich, Dario, wollte schon als kleines Kind Spiele erstellen. Anfangs waren es Brettspiele mit Papierfigürchen und farbigen Papier, später lernte ich auf Scratch zu programmieren und jetzt will ich mich der nächsten Herausforderung stellen. Mich fasziniert die Erstellung von Fantasiewelten aus dem Nichts. Als passionierter Gamer hinterfrage ich wie ein bestimmtes Element oder Objekt gemacht wurde. Bisher war ich ein Consumer, aber jetzt möchte ich mal ein Spiel erstellen und entscheiden können, welches meinen Vorstellungen entspricht. Ich schätze es ein Team zu haben, in dem die Arbeiten aufgeteilt werden können, denn der Auftrag ist schwierig und zeitaufwendig.

MARTIN

Die Idee für ein Video-Spiel kam von Dario, aber schon vor der Lehre hatte ich vor ein Video-Game zu erstellen, doch ich war viel zu faul und wollte nicht lernen. Ich spiele hauptsächlich Schiesserei Spiele oder sogenannt First Person Shooter (FPS) und ich denke so eins zu erstellen wäre etwas zu kompliziert, also könnte ich es durch 2D vereinfachen. Es ist mir nie eingefallen, dass es so schwierig ist, ein game zu erstellen, jetzt habe ich viel mehr Respekt vor Game Developers und ich hoffe, dass meinen neuen Kenntnissen mich beim Gamen helfen könnten.

ARYAN

Aryan war von Anfang an ein Mitglied des Teams. Seine Aufgabe war es, uns zu unterstützen. Er erstellte das Titelblatt und half bei den Entscheidungen. Leider ist er nun nicht mehr an dieser Schule. Wo Martin und ich beim technischen Teil sehr involviert waren, war es für Aryan nie spannend zu programmieren oder gross mitzumachen. Wir gaben ihm die Aufgabe das Design zu machen, aber er kam nicht weit. Zusammenfassend sind wir dankbar für Aryans Hilfe und bringen das Projekt in Gedanken für ihn über die Ziellinie.

IPERKA

I (INFORMIEREN)

PROJEKTAUFTRAG

Bezeichnung	Videospiel erstellen
Entwicklerteam	The Tough Shells: Dario Kulici, Martin Sorrenti
Auftraggeber	M431: Thomas Kälin
Projektbeginn / Ende	16.05.2024 / 04.07.2024
Ziel	Ein Arena Shooter Video Spiel erstellen, mit Gegner KI
Beschreibung	Arena Shooter Spiel, indem man Gegner mit Messer töten muss und dafür Punkte kriegt, der Spieler kann nach X Punkte das Spie verlassen oder nachdem er stirbt.
Ergebnis	Ein spielbares Spiel, welches man über GitHub herunterladen kann

BESCHRÄNKUNGEN / RESSOURCEN

Wir mussten uns auf einige wichtige Entscheidungen einigen.

Unser Auftrag ist es, mit der Godot Game Engine ein einfaches, spielbares Spiel zu erstellen. Wenn alles passt, dies auch zum Herunterladen bereitstellen. Wir denken das Spiel soll nicht zu schwierig zu programmieren sein, jedoch muss es auch beim Spielen Spass machen. Dafür brauchen wir am genug Zeit für die Einarbeitung und Zielsetzung. Da wir noch keine Erfahrung in diesem Gebiet haben, können wir auch nicht 100% sagen, was wir wann und wie machen müssen. Was wir aber können ist uns eigene Kriterien zu setzen, nach denen wir das Spiel programmieren wollen.

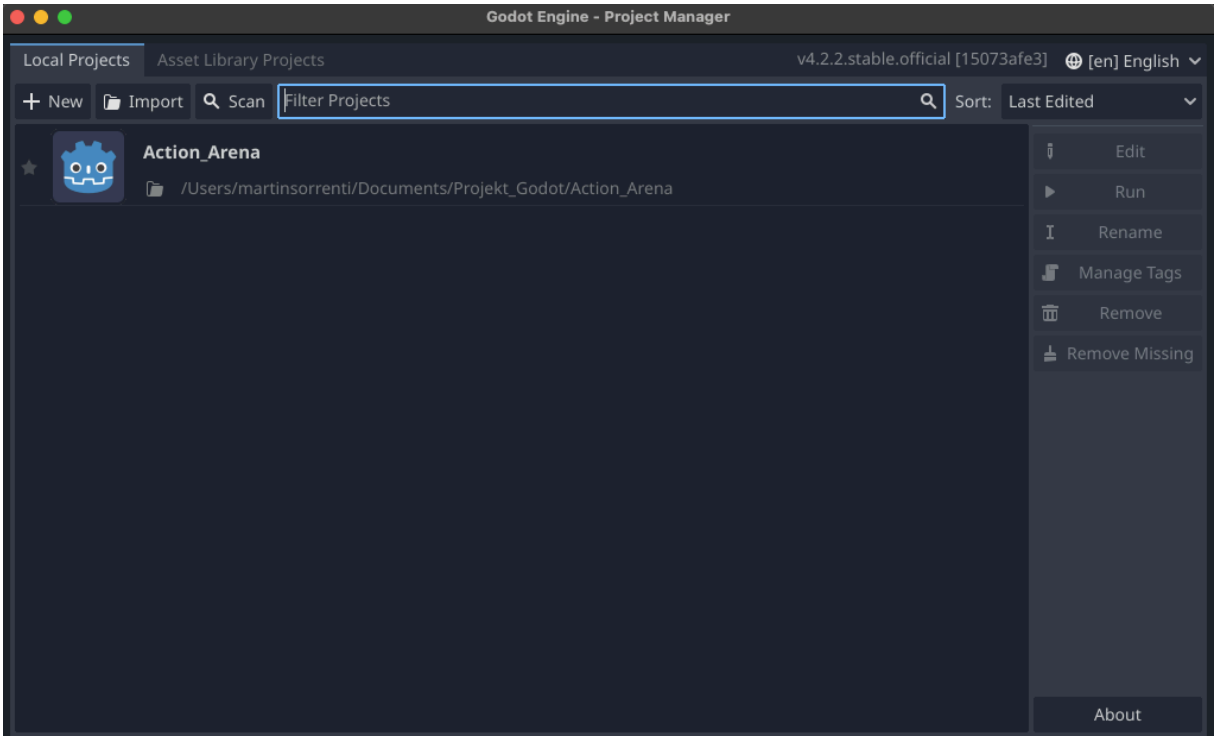
Kriterien:

- 2D (3D wäre zu aufwendig und würde zu viel Zeit in Anspruch nehmen)
- Shooter (Wir spielen alle gerne Shooterspiele, weshalb wir selbst einen erstellen wollen.)
- Gegner KI (Es soll logischerweise einen Gegner haben, den man umnieten kann.)
- Funktionierender Spieler (Leben-, Schiess- & Punktesystem)
- Keine Bugs (Unerwartete Fehler, die die Immersion kaputt machen)
- Flüssiges Gameplay (Das Spiel soll auf jedem Gerät der Klasse spielbar sein.)

Das Spiel haben wir nach diesen Kriterien gegründet und versucht zu verbessern.

Was ist eine Game Engine?

Damals programmierte man von Null. Das heisst man musste die Gravitation, Lichtverarbeitung oder die allgemeine Verarbeitung der Daten selbst programmieren. Heute gibt es dafür Game Engines, die eine Grundbasis für ein Projekt darlegen. Funktionen wie fortgeschrittene Lichtverhältnisse, Rendering, Audiohandling, Networking oder Collisionsystems sind in guten Game Engines vorhanden und nutzbar. Komplexere Game Engines wie zum Beispiel die Unreal Engine von der Firma Epic Games haben sehr fortgeschrittene Technik verbaut.



Fragen

Fragen an Auftraggeber	Antworten von Auftraggeber
Ist das Produkt oder das Verständnis von IPERKA wichtiger?	Das Verständnis von IPERKA ist wichtiger.
Wie komplex soll bzw. darf das Spiel sein?	Solange es in den Zeitrahmen passt ist viel Spielraum vorhanden.

P (Planen)

Bei der Planung ist es wichtig, dass alle Aufgaben klar und deutlich erfasst sind. Dafür haben wir ein Excel Dokument und diese Dokumentation benutzt. Das Aufwendigste an unserem Projekt ist einzuschätzen wie viel in dieser Zeit möglich ist. Anfangs wollten wir einen komplexen Shooter mit Networking (Multiplayer) erstellen, dann stellte sich heraus, dass die Zeit nicht reicht. Wir müssen uns auf eine simplere Version des Spieles einigen.

Hier ist unsere Metaeben, die wir auf Excel erstellt haben. Das sind die groben Aufgaben und die jeweilige Zeiteinteilung.

To Do	Wer	Zeit	Verlinkung
Einarbeitung in Materie	Martin, Dario	10h	
Visuals und Assets erstellen	Aryan, Martin, Dario	24h	
Gameidee erstellen	Aryan, Martin, Dario	3h	
Basis des Spieles erstellen	Martin, Dario	1h	
Spielersystem erstellen	Martin	3h	
Waffen kreieren	Dario	2h	
Tilemap erstellen	Martin	1h	
Gegner KI aufsetzen	Martin	4h	
Lebensystem erstellen	Martin	2h	
Looting System erstellen	Dario	3h	
Bugfixing und Polishing	Martin, Dario	10h	
Dokumentation	Aryan, Martin, Dario	10h	Dokumentation

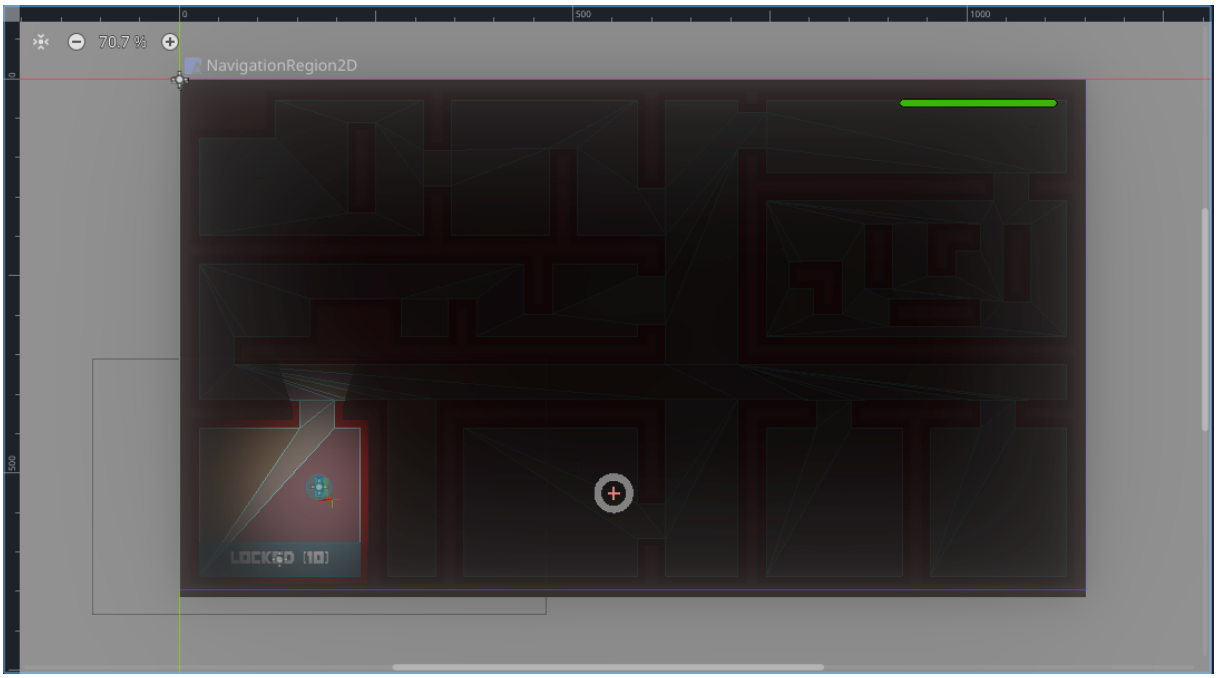
Die Ressourcen haben wir auch festgelegt. Wir haben wenig Ressourcen, dafür ein komplexes Zusammenarbeiten. Auf Obsidian machen wir Notizen, auf Godot das Spiel, auf Word die Dokumentation und mit Libresprite die visuellen Aspekte des Spiels.

Ressourcen	Wer
Godot Engine	Dario Kulici
Libresprite	Martin Sorrenti
Obsidian	Martin Sorrenti
Word	Dario Kulici

Am Schluss mussten wir endlich das fertige Spiel programmieren. Bisher sammelten wir Erfahrung und deshalb ist hier die Einteilung, wer was macht. Die Zeiteinteilung schweift von der oberen To Do Liste ab, da dies das schlussendliche Projekt ist, ohne die Einarbeitungsphase.

Was?	Wer?	Zeit
Player	Dario	2 h
Maingame (Tilemap, Extraction System)	Martin	5 h
Menüs (UI, Startmenu)	Dario	1h
Interactions (Waffen, Bandagen, usw.)	Martin	4 h
AI	Dario	3h

Es brauchte auch Zeit für die Idee des Spiels. Zuerst wollten wir ein Überlebensspiel erstellen, dann änderten wir die Idee immer mehr und schlussendlich entschieden wir uns für einen Arena Shooter. Das Prinzip ist ganz einfach. Es gibt einen Spieler und gegnerische Kreaturen. Das Ziel des Spielers ist es so lange wie möglich am Leben zu bleiben, während die Kreaturen den Spieler angreifen. Je länger der Spieler überlebt, desto höher ist seine Punktzahl. Im Spiel gibt es dann Bonusinhalte, die teilweise Vorteile aber auch Nachteile für den Spieler schaffen.



E (Entscheiden)

Nach vielen Überlegungen und Abschätzungen entschieden wir uns einen Arena Shooter zu machen. Wir notierten die, zu erledigenden Aufgaben, und wie viel Zeit bzw. Ressourcen wir in die dementsprechenden Aufgaben stecken wollten. Für die Einarbeitung z.B. schrieben wir 10h auf. Die Idee, den Multiplayer einzufügen, mussten wir auch lassen, weil es zu zeitaufwendig und schwierig wäre.

Wir hatten viele Ideen, was man hätte machen können, aber mussten uns auf weniger Inhalt beschränken.

Wahl der Game Engine

Da wir Anfänger sind entschieden wir uns für die Godot Engine. Sie ist simple gehalten, hat aber viele Features und ist im Ganzen sehr gut.

Wichtige Aspekte unseres Spieles

Es war uns von Anfang an wichtig, dass unser Spiel immersiv wird. Immersion ist das Gefühl, wenn man tief in eine Sache vertieft ist und an nichts anderes denkt. Bei einem guten Videospiel ist es auch das Gefühl, das süchtig macht. Um die Immersion beizubehalten sind verschiedene Faktoren zuständig. Bugs oder Fehler, die dem Spieler auffallen, zerstören die Immersion sofort. Uns ist es deshalb sehr wichtig, dass solche Fehler nicht in unserem Spiel vorkommen. Ein weiterer Punkt sind Emotionen. Wenn der Spieler eine Angst spürt, um die nächste Ecke zu gehen, ist er fokussierter und konzentrierter.

R (Realisieren)

Dank Godots einfachem GUI (Graphics User Interface) war es sehr einfach klarzukommen und somit konnten wir auch vieles Testen und immer neue Projekte erstellen, was sehr wichtig ist für das Lernen des Godot Syntaxes. Es war auch schwierig die verschiedenen Scripts und Szenen zusammen zu verbinden, damit es verständlich und effizient ist. Bei anderen Game Engines wie z.B Unity oder die Unreal Engine werden Sprachen wie C++ oder C# verwendet, die um einiges schwieriger sind zu lernen. Godot ist gerade die perfekte Mitte zwischen schwachen Game Engines (Big World Engine, Cry Engine) und Super Game Engines (Unity, Unreal Engine).

```
extends CharacterBody2D

@export var speed: float = 170
@export var rotation_speed: float = 10

var health = 100.0

signal life(health)

func _physics_process(delta):
    var direction = Input.get_vector("left", "right", "up", "down")
    velocity = direction * speed

    look_at(get_global_mouse_position())
    move_and_slide()


func take_damage(enemy_damage):
    health -= enemy_damage
    life.emit(health)

    if health <= 0:
        get_tree().quit()
```

Anfangs experimentierten wir mit verschiedenen Ideen und Entwürfen. Das Wichtigste war, erste Erfahrungen zu sammeln, damit wir in Zukunft effizienter arbeiten und mehr Wissen anwenden konnten. Insgesamt haben wir 4+ Projekte erstellt und angepasst damit wir eine Basis für das eigentliche Spiel haben. Mit Projekt meinen wir «einen Ordner» mit dem Spielinhalt auf Godot. Das Ziel war es am Schluss ein Projekt zu erstellen und dort das ganze Wissen zu einem fertigen Spiel zusammenzutragen. Was uns sehr viel geholfen hat, sind YouTube Tutorials und unsere Zusammenarbeit. Uns ist aufgefallen, dass die gegenseitige Motivation sehr entscheidend ist. Wenn wir uns nicht gegenseitig gefördert und motiviert hätten, hätten wir kein tolles Spiel wie jetzt.

Meilensteine

Wir habe uns auch ein paar Meilensteine gesetzt:

1. Godot kennenlernen
2. Unser erstes Projekt kreieren
3. Einen kontrollierbaren Spieler / Character zu erstellen
4. Eine Waffe, die schiesst, zu implementieren
5. Einen Gegner zu erstellen
6. Eine Mappe / Spielarena zu gestalten
7. Einen Spawner für den Gegner zu bauen
8. Leben / HP implementieren
9. Score System erstellen
10. Das Spiel installierbar auf GitHub hochladen

Die meisten haben wir auch geschafft, jedoch konnten wir das Spiel nicht auch GitHub hochladen, weil die Datei viel zu gross war für eine gratis GitHub Repository.

K (Kontrolle)

Wir definierten anfangs einige Tests, um zu kontrollieren, wie nah bzw. ob wir unser Ziel erreicht haben. Uns ist es wichtig, dass das Spiel wenig Bugs hat und flüssig funktioniert. Das beinhaltet eine effiziente Programmierung und ein gutes Zusammenspiel zwischen den Szenen. Die Hauptschwierigkeit dabei, ist es Daten zwischen Szenen mithilfe von Signalen zu bewegen und interpretieren zu lassen. Bugs sind «Käfer», die in Spielen auftauchen und das Erlebnis beeinträchtigen. Meistens sind es nicht vorgesehene Lücken, die beispielsweise unmögliche Aktionen, wie bspw. Durch eine Wand gehen, möglich machen. Um diese Bugs zu finden, benötigt es Zeit fürs Testen.

Tests

Hier eine Übersicht von den definierten Tests:

INDEX	TESTNAME	TESTFRAGEN
1	Bugkontrolle	Hat das Spiel Bugs? Sind es schlimme Bugs?
2	Performancekontrolle	Läuft das Spiel flüssig? Wie verhält es sich auf verschiedenen Geräten? Wie viel FPS (Frame per Second) sind möglich?
3	Spielspasskontrolle	Macht es Spass? Gibt es genug Faktoren für Glückgefühle?

Bisher gingen die ersten beiden Tests sehr gut. Das Spiel hat keine Bugs und die Performance ist super, da wir keine aufwändigen Texturen nutzen. Unserer Vermutung nach liegt es an der Einfachheit des Spiels. Dadurch, dass wir wenige Szenen und nicht aufwändige Texturen nutzen, ist das Spiel auch weniger anfällig für Bugs oder Fehler. Unser Spiel ist nicht mehr als 150 MB gross, weswegen es auch wenige Abhängigkeiten gibt, in denen es Bugs haben könnte. Wie bereits erwähnt hat unser Spiel einfache Texturen, die nicht aufwendig zu rendern sind. Dies verschafft uns den Vorteil, dass es auf allen Geräten spielbar ist ohne jegliche Performanceprobleme.

Testing

Um den Spielspass zu testen, brauchen wir andere Personen, die ein Feedback geben können, wie sie das Spiel finden und welche Verbesserungsvorschläge sie haben. Dafür haben wir drei Kandidaten ausgesucht.

KANDIDAT	VERBESSERUNGSVORSCHLÄGE	POSITIVE RÜCKMELDUNG	REAKTION AUF FEEDBACK
DAVID PETER	Verbesserungsvorschläge • Eine Art Animation bei Verletzungen • Kill-Anzeige (wie viele Tötungen) • Heil Gegenstände auf der Landkarte • Mehr Gegenstände wie Wurfmesser, Molotov etc.	• Einwandfreie Performance • Keine Bugs	• Mehr Animationen hinzufügen • Kill-Anzeige hinzufügen • Bonusgegenstände auf der Landkarte hinzufügen (bspw. Heilgegenstand) • Wir können nicht mehr Waffen implementieren, da es zu aufwendig ist.
ILAI LUISE	• Mehrere Wellen mit mehr Zombies • Mehr Effekte (bspw. Lichteffekt bei Schuss) • Mehr Details (Objekte)	• Keine Bugs • Flüssige Performance	• Effekte und mehr Details im Spiel können wir gut realisieren, da es nicht zu aufwendig ist • Mehrere Wellen einzubauen ist ein technischer Aspekt, der höchstwahrscheinlich zu viel Zeit braucht, weswegen wir das nicht einbauen können.
ALDIN SAHINOVIC	• Zombies langsamer machen	• Super Performance (etwa immer 60FPS) • Keine Bugs	• Wir werden auf jeden Fall Balanceänderungen durchführen, damit das Spiel fair ist.

**alle aufgeführten Reaktionen auf das Feedback werden wir nicht im Zeitrahmen des Moduls durchführen, da die Zeit zu knapp ist. Der Sinn ist es, die Anpassungen nach dem Modul weiterzuführen.*

A (Auswertung)

Stakeholder

Entwicklerteam: Dario Kulici und Martin Sorrenti (ursprünglich auch Aryan Awal)

Auftraggeber: Thomas Kälin (M431)

Tester: David Peter, Ilai Luise & Aldin Sahinovic

Verlauf

Informationsphase: Das Team informierte sich über Game Engines und entschied sich für Godot.

Planungsphase:

Ursprüngliche Idee eines komplexen Shooters mit Multiplayer und Extraktion. Anpassung zu einem simpleren Arena Shooter aufgrund von Zeitbeschränkungen.

Entscheidungsphase:

Festlegung auf einen 2D Arena Shooter. Auswahl der Godot Engine als Entwicklungsumgebung.

Realisierungsphase:

Erstellung mehrerer Testprojekte zum Erlernen von Godot. Umsetzung von 9 von 10 geplanten Meilensteinen

Kontrollphase:

Durchführung von Tests wie Bugfixing, Performancetest und Spielspasskontrolle. Einholen von Feedback von externen Testern

Produkt

2D Arena Shooter, indem der Spieler einen Gegner mit einem Messer töten muss und somit Punkte sammeln. Einfache Grafik und Texturen für gute Performance. Keine schwerwiegenden Bugs. Spielgröße unter 150 MB.

ERFAHRUNGEN

Zeitmanagement:

Erkenntnis, dass komplexe Features wie Multiplayer den Zeitrahmen sprengen würden. Notwendigkeit, Ideen zu reduzieren und sich auf das Wesentliche zu konzentrieren.

Technische Erkenntnisse:

Godot als benutzerfreundliche Engine für Anfänger. Herausforderungen beim Verbinden verschiedener Skripte und Szenen.

Teamarbeit:

Bedeutung gegenseitiger Motivation und Unterstützung. Schwierigkeiten bei der Integration von Teammitgliedern mit unterschiedlichem Interesse (Aryan).

Spielentwicklung:

Erkenntnis über die Komplexität der Spieleentwicklung. Wichtigkeit von Immersion und Bugfreiheit für ein gutes Spielerlebnis.

Projektmanagement:

Praktische Anwendung der IPERKA-Methode. Bedeutung von regelmäßigem Testing und Feedback.

Insgesamt haben wir wertvolle Erfahrungen in der Spieleentwicklung und im Projektmanagement gesammelt, trotz einiger Herausforderungen und notwendiger Anpassungen im Projektverlauf.