

/* Idee und Umsetzung des Projektes */

</Projektidee>

Ziel ist es, den interaktiven Entscheidungsbaum so zu konzipieren, dass die Nutzendenaktionen eine Kommunikation zwischen Client und Server initiieren und auf diese so gespeicherten Daten im weiteren Spielverlauf zugegriffen werden kann. Da es sich nicht um ein kompetitives Spiel handelt, bei welchem sich die Interaktionen auf Grundlage eines Zusammenspiels bzw. des gegeneinander Spielens einbauen lassen, entstand die Idee einer Statistik. Diese wird das zentrale neue Element des Spiels, was bedingt, dass alle weiteren Interaktionen dazu dienen sollen, die für die Statistik benötigten Daten während des Spielens abzuholen.

</User Interface>

Vor Beginn des Spiels werden die User aufgefordert ihren Namen einzugeben, ohne diesen das Spiel nicht gestartet wird, da es im Hintergrund die initiiierende Funktion ist, um abzugleichen, ob es sich um einen rückkehrenden User handelt oder ein neues Spiel angelegt werden soll. Zuoberst befindet sich die ausklappbare Infobox, mit einer Spielanleitung sowie die Auswahloption für die Spielfigur. Mittig liegt das quadratische Spielfeld und darunter befindet sich das eigentliche Quiz, wo die klickbaren Fragen und Antworten und am Spielende auch das Resultat angezeigt wird. Ausserdem ist es zu jedem Zeitpunkt möglich das Spiel nochmal ganz neu zu starten. Beim Abschluss des Spiels kann man sich die Statistik anzeigen lassen.

</Visualisierung>

Die Visualisierung des Spielfelds besteht aus drei übereinander gelagerten Canvas Elementen, welche im Spielverlauf je nach Frequenz individuell bearbeitet und bereinigt werden. Die Visualisierung der Statistik am Ende des Spiels wird mit der JavaScript-Bibliothek D3 realisiert. Diese Architektur hält den CSS-Code sehr schlank, da das Stylesheet im Grunde nur für das Layout des Canvas und der interaktiven Elemente wie Buttons oder Toggles benötigt wird.

</Serverinteraktion>

Auf Grundlage einer API-Definition wurde ein Server angelegt, was nun neue und verbesserte Funktionalität ermöglicht. Die Inhalte und Verknüpfungen zwischen den einzelnen Fragen und dazugehörigen Antwortmöglichkeiten wurden aus dem Code ausgelagert auf den Server abgelegt und werden somit beim Auslösen der entsprechenden Fetch-Methode aufgerufen. Dasselbe gilt für die Inhalte der Resultate. Zentrale Interaktionen betreffen das Anlegen eines neuen Spiels bzw. das Abrufen eines bestehenden, sowie das Aktualisieren eines laufenden Spiels. Diese Interaktionen sind wiederum parallel auch mit der Statistik verknüpft, welche genauso mit den entsprechenden Daten befüllt wird.

</Datenstruktur>

Der JavaScript-Code beinhaltet die Funktionen zur Visualisierung des Canvas inkl. der Animation während des Spiels, wie auch der dazugehörigen Schaltflächen und das Einblenden der unterschiedlichen Ansichten je nach Spielfortschritt. Ausserdem enthält der JavaScript-Code die asynchronen Funktionen zur Serverkommunikation und die Elemente des Gameplay. Verschiedene Funktionen koordinieren dabei die Spielführung, stellen den korrekten Ablauf sicher und rufen dafür die nötigen Daten vom Server ab bzw. legen sie dort ab. Auch das Abrufen und Anlegen der Statistik ist in JavaScript programmiert. Das Layout wurde in einem

simplen HTML-Dokument definiert, welches im Body die Infobox, die Canvas-Elemente, verschiedenen Buttons und ein paar wenige Textelemente enthält. Obschon nur ein HTML-Dokument erstellt wurde, wurde zwecks Übersichtlichkeit die Formatierung der HTML-ELEMENTE in eine CSS-Datei ausgelagert. Die Visualisierung der Statistik konnte mit dem Einbinden des Scripts der D3-Bibliothek innerhalb des HTML-Dokuments realisiert werden.

/* Erläuterung des Source Codes */

</Abschnitt>	</Elemente>	</Beschreibung>
1 Canvas - Visualisierungen	const size, const corra, const corrb, const coordinates, function clear(layer), function vac(a,b), function dots(), function loadlines(knots), function dot(a,b,dotsize,colour), function line(a,b,c,d), function maindots(a,b,type), function drawdown(p1,p2,i,j), function drawside(p1,p2,i,j),	Size ist eine universelle Masseinheit, auf welche alle Grössen und Abstände Bezug nehmen. Sie stellt sicher, dass die Canvas-Visualisierung mittels Anpassung von einer Variable vergrössert oder verkleinert werden kann. CorrA und CorrB sind reine Kalibrierungen, diese können angepasst werden um dots und Spielfigur aneinander auszurichten. Das Array mit den Koordinaten der Punkte, wurde im app.js belassen, da diese von Beginn an geladen werden. Die weiteren Funktionen enthalten Instruktionen zum Zeichnen von Visualisierungselementen bzw. Animationen des Spielfelds.
2 Inter- aktivität	function gameinfo(i), function fade(stringid, type), function loadsite(site),	Diese Funktionen steuern das Ein- und Ausblenden und Verbergen der Schaltflächen. Eine zentrale Funktion ist loadsite(site), weil diese eine Abfrage vornimmt, um den Spielfortschritt zu ermitteln und aufgrund dessen die dazugehörige Ansicht lädt.
3 Gameplay	const currentgame, function validate(), function checkGame(), function createGame(count), function reloadGame(result), function initialise(), function loadquestion(data), function choice(x), function reload(), function loadresult(result), function restart()	Eine zentrale Variable ist currentgame, da sie die erforderlichen Angaben zu jedem Spiel speichert, welche als solche auf dem Server abgelegt werden. Die aufgeführten Funktionen sorgen für einen koordinierten Spielablauf, indem sie verschiedenste Serverinteraktionen auslösen. Einerseits geht es darum zu prüfen, ob ein Spiel bereits besteht oder ob ein neues angelegt werden muss, welches dann gestartet wird. Choice nimmt die Antworten der Nutzenden entgegen und verknüpft diese mit loadquestion, um die richtige Frage/Antwort-Kombination oder das Resultat zu laden.
4 Server - kommunikation	async function fetcher(method, directory, data)	Es wurde eine zentrale asynchrone Funktion programmiert, welche alle eingesetzten fetch-Befehle beinhaltet. Diese Funktion wird über die Gameplay-Funktionen ausgelöst bzw. greifen die Gameplay-Funktionen mithilfe der fetcher-Funktion auf die benötigten Daten zu bzw. geben die zu speichernden Daten an den Server weiter. Somit ist es möglich eine Funktion und deren Befehle für mehrere identische Abfragen zu nutzen, deren Daten dann aber trotzdem an verschiedene Funktionen weitergegeben werden können.
5 Daten - visualisierung	function visualize(result), d3.select().append, d3.arc	Diese Funktion verarbeitet die Daten, welche bei der Abfrage der Gesamtstatistik, als Array herausgegeben wurden, zu Substatistiken. Deshalb kann auf die Substatistiken mithilfe des Index zugegriffen werden. Die Daten werden mittels d3 in ein Diagramm verarbeitet

/* Reflexion */

</Highlights>

Obschon wir auf dem Weg zum Endresultat diverse Umwege, Unterbrechungen und Rückschritte in Kauf nehmen mussten, ist der Projektausgang dennoch als positiv zu verbuchen. Das gesteckte Ziel, die Umsetzung der Projektidee, wie sie von Beginn an Bestand hatte, konnten wir realisieren. Wir haben unser Ziel in einem Bereich sogar übertroffen, war es zu Beginn doch noch gar nicht angedacht, die Statistik mittel d3.js zu visualisieren, sondern als reine Zahlen auszugeben. Beim Formulieren der Projektidee schien uns das bloße Umsetzen von Serverinteraktionen bereits Herausforderung genug. Ein weiteres Highlight ist die Verbesserung einiger Abschnitte des bestehenden Codes. Abläufe wurden nochmals überdacht, so dass das Programm nun viel schlanker und kompakter daher kommt, was das Ganze weniger fehleranfällig, skalierbarer und kompatibler macht. Dies hat sich v.a. bei der Verknüpfung zum Server gezeigt, die dann für verschiedene Interaktionen ähnlich gestaltet werden konnte. Trotzdem war es ein Highlight zu sehen, dass der bestehende Code von der Logik her bereits viele Umsetzungen vereinfachte. Obschon die Applikation nun entscheidend komplexer ist und mehr Funktionen hat, ist der Code, dank der asynchronen Abläufe, nun nicht bedeutend länger.

</Lowlights>

Der Start in das Projekt erwies sich als zäh. Obwohl wir im Nachhinein sagen können, dass die Projektidee realistisch formuliert war und auch die Funktionalitäten enthält, welche tatsächlich umgesetzt wurden, fiel es uns eher schwer den Weg dahin uns vorzustellen. Für die API-Definition zu Beginn waren bereits mehrere Sitzungen in der Gruppe nötig, da alle Mitglieder gleich wenig Vorwissen dazu mitbrachten und ein Teammitglied im vorherigen Semester an einem anderen Projekt arbeitete. Im Verlauf zeigte sich dann auch, dass die Funktionalität einer API erst im letzten Drittel des Projekts verstanden wurde, als man damit konkret gearbeitet hatte, weshalb die Definition zu einem späten Zeitpunkt nochmals überarbeitet und ausgebaut wurde. Eine Statistikabfrage, welche die aktuell spielenden User ausgeben würde, konnte nicht realisiert werden. Aus Zeitgründen wurde auf die Ausbesserung der Textcodierung von Umlauten auf dem Server und kosmetischen Anpassungen des CSS bzw. des Designs allgemein verzichtet. Es wäre hier sicher eine Überlegung Wert gewesen, ein JavaScript Framework einzusetzen.

</Learnings>

Viele Probleme traten wie beschrieben v.a. in der ersten Hälfte auf und konnten zu diesem Zeitpunkt auch nicht vollständig gelöst werden, weil sie einerseits von den grossen Wissenslücken im Bereich Webprogrammierung und andererseits von der Struktur der Spielvorlage aus dem vergangenen Semester stammen. Wir hätten diese schwierige Projektphase schneller hinter uns lassen können, wenn wir der Projektidee mehr Vertrauen geschenkt hätten und trotz der Unsicherheiten gezielter mit der Umsetzung begonnen hätten. Die meisten Verständnisschwierigkeiten hatten sich dann in der Umsetzung zu lösen begonnen. Ausserdem war es lehrreich zu sehen, wie sich Abläufe, welche im Vorprojekt schwierig zu lösen waren, nun mit einer Serverinteraktion viel einfacher programmiert werden können.

</Fazit/{Next Steps}>

Die Idee an sich konnte wie geplant realisiert werden. Da im Spiel keine persönlichen Daten abgefragt (bspw. kann der Username frei erfunden werden) und gespeichert werden und nur die aktuellen Spieldaten übertragen werden, so dass die Spieldaten unter den Usern nicht gegenseitig einsehbar sind, wurden zum Schutz vorerst keine Massnahmen ergriffen. Die Fehlerbehandlung konnte für einen HTTP-Status-Code erreicht werden, welcher Dank der Struktur des Programmcodes ein Neuladen der Seite auslöst und die User trotzdem nahtlos weiterspielen lässt.

Da die Funktionen sehr unterschiedliche Rückgabewerte aufweisen und häufig als Kette von Funktionen einander gegenseitig auslösen, ergab ein Testing mittels Testfunktionen in diesem Zusammenhang wenig Sinn. Die Funktionalität der Vorgänge konnte erfolgreich mit Konsolen-Mitteilungen überprüft werden.

Im weiteren Verlauf sollten die Fehlerbehandlungen auf weitere Status-Codes ausgeweitet werden. Das Design könnte weiter verfeinert und optimiert werden und die Vergabe der IDs, könnte ebenfalls anders gestaltet werden, weil das Absuchen von bestehenden Daten je öfters das Spiel gespielt wurde umfangreicher wird.

/* Quellen und Pfade */

</Abschnitt>	</Beschreibung>	</Link>
CSS, HTML, JavaScript	Verschiedenste Tutorials konsultiert, jedoch kein Code 1:1 übernommen.	https://www.w3schools.com/
CSS "Toggle"	Funktionsweise grösstenteils übernommen, Design etwas angepasst.	https://codepen.io/Unleashed-Design/pen/aYOZyW
Serverinteraktion	Grundlegender Aufbau der Fetch-Befehle	https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch
Arrow-Functions	Aufbau einer Arrow-Funktion zum Einsatz bei den Fetch-Befehlen	https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Functions/Arrow_functions
d3.Pie	Gibt die Statistik grafisch als Kuchendiagramm aus.	https://www.youtube.com/watch?v=L5GXOdt2uOc (Part I und II)
</Pfad zum Repository> https://github.com/hilgeann/ProjektWEBP		

/* Eidesstattliche Erklärung */


Ich erkläre hiermit, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und erlaubten Hilfsmittel benutzt habe. Alle Stellen, die wörtlich oder sinngemäss aus Quellen entnommen worden sind, habe ich als solche gekennzeichnet. Mir ist bekannt, dass andernfalls die Hochschulleitung zum Entzug der aufgrund meiner Arbeit verliehenen Qualifikation oder des für meine Arbeit verliehenen Titels berechtigt ist.



Basel, 03. Juni 2022 (A. Hilgert)



Bern, 03. Juni 2022 (R. Jevdenic)



Grenzach-Wyhlen, 03. Juni 2022 (T. Siefert)