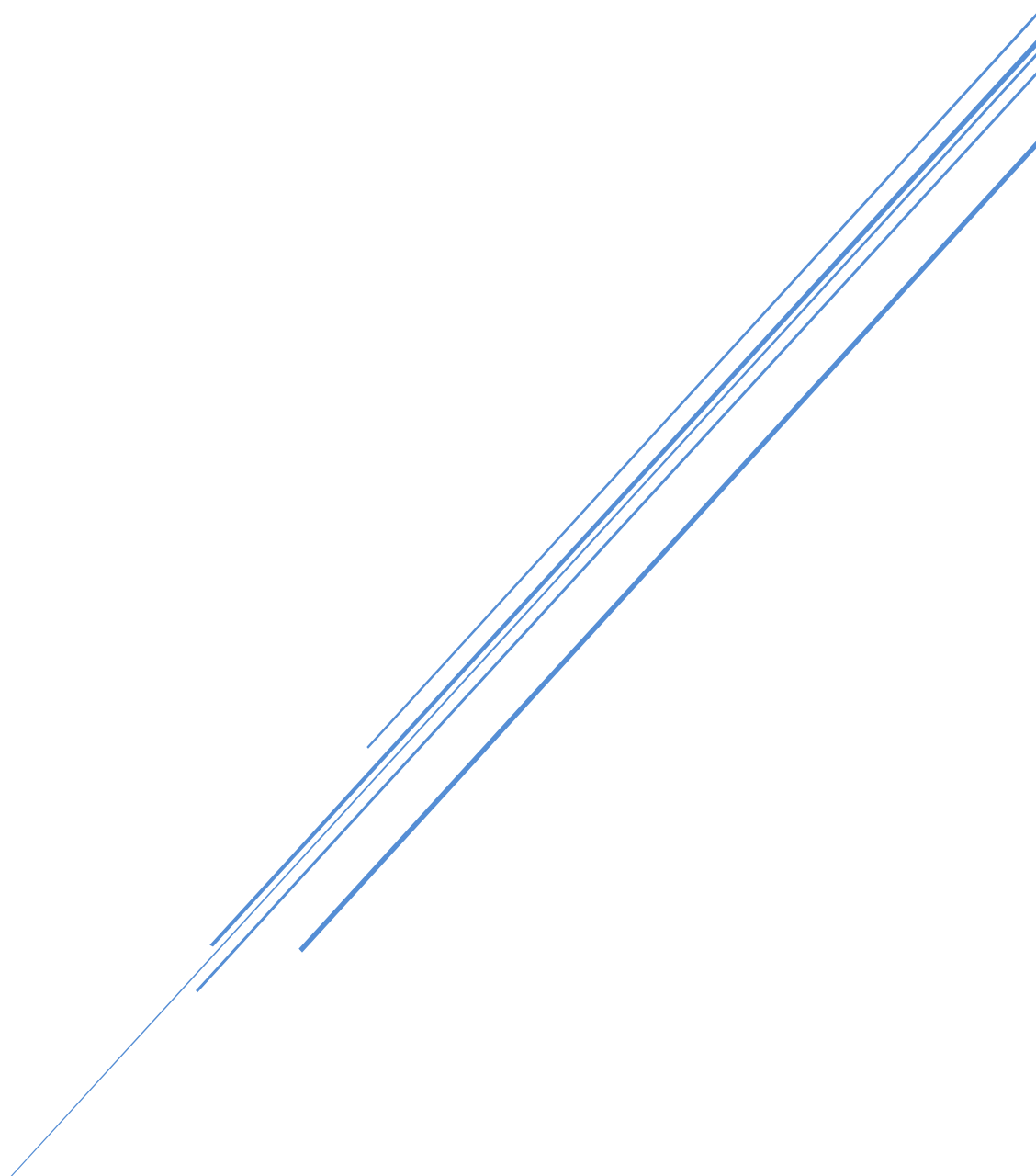




Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Eidgenössisches Departement für
Wirtschaft, Bildung und Forschung WBF
Information Service Center WBF ISCeco

JUMP AND RUN «MODIS»



ISCeco
Lehrlingswesen



Inhaltsverzeichnis

1	Kurzfassung	2
2	Organisation Projektergebnisse	2
3	Initialisierung	3
3.1	Projektziele	3
3.2	Variantenentscheid	3
3.2.1	Kriterien und Gewichtung	3
3.2.2	Skala	4
3.2.3	Variante 1: PixiJS	4
3.2.4	Variante 2: Phaser	5
3.2.5	Variante 3: GDevelop	5
3.2.6	Auswertung	6
3.3	Umfeld	6
3.4	Abgrenzungen	6
4	Konzept	7
4.1	Technische Spezifikation	7
4.2	Fachspezifikation	7
4.3	Use Case	10
4.4	Benutzeroberflächen	15
5	Realisierung	15
5.1	Verweisung auf Beispielprojekt	15
5.1.1	Genutzte Extensions	15
5.2	Entwicklung mit GDevelop	15
5.2.1	Variablen	15
5.2.2	Event-System	16
5.2.3	Behaviors und Animations	16
6	Abbildungsverzeichnis	17
7	Tabellenverzeichnis	17
8	Quellenverzeichnis	17
9	Glossar	18



1 Kurzfassung

Der Lehrverbund darf an der diesjährigen BAM teilnehmen, um die Informatik zu präsentieren. Dazu soll ein Spiel entwickelt werden, welches die Informatiklehre und deren Werdegang in einem spielerischen Rahmen aufzeigt. Dabei soll zunächst ein simples 2D Jump and Run-Spiel bereitgestellt werden. Die Entwicklung dieses Spiels erfolgt über GDevelop¹.

2 Organisation Projektergebnisse

Die Projektergebnisse werden in GitLab im entsprechenden Repository untergebracht. Damit die Lernenden, die nächstes Jahr ins ISCeco kommen, bereits einen Überblick vom Projekt haben und mitwirken können, werden die Projektergebnisse ebenfalls in einem entsprechenden Repository in GitHub gesichert.

¹ [GDevelop](#)



3 Initialisierung

Nun werden die Rahmenbedingungen sowie die Grundlagen für das Projekt definiert und in den nachstehenden Unterkapiteln festgehalten.

3.1 Projektziele

Das Ziel ist es ein 2D Jump and Run-Spiel bereitzustellen, welches die Informationen über die Informatiklehre inklusive der Module, Üks und IPA spielerisch vermittelt. Für die bevorstehende BAM 2025 steht zunächst die Spielentwicklung für das Handy im Fokus.

3.2 Variantenentscheid

Um schnellstmöglich ein solches 2D-Spiel zu entwickeln, wird eine Game Engine benötigt. Da im IS-Ceco noch keine Erfahrungen mit Game Engines erfolgten, müssen Varianten verglichen und eine Wahl getroffen werden.

3.2.1 Kriterien und Gewichtung

Bevor die einzelnen Varianten verglichen werden, müssen jedoch die Kriterien, wonach sie bewertet werden, sowie deren Gewichtung festgelegt werden. Dabei handelt es sich um Kriterien, die für dieses bevorstehende Projekt in der aktuellen Situation (Ende Juni 2025) von Bedeutung sind.

Für die Gewichtung wird der Zahlenwert 100 auf die kommenden vier Kriterien verteilt. Somit würde jedes Kriterium eine Gewichtung von 25 haben. Wenn nun ein Kriterium relevanter ist als ein anderes, so wird dieses eine höhere Gewichtung erhalten. Jedem einzelnen Kriterium wird eine solche Gewichtung zugewiesen. Die Gewichtung wird abhängig von der Relevanz des entsprechenden Kriteriums für das Projekt bestimmt.

Kriterium	Beschreibung	Gewichtung
Open-Source	Da kein Budget für dieses Projekt besteht, darf die Game Engine nicht kostenpflichtig sein. Alle benötigten Features der Game Engine sollten somit Open-Source sein.	30
Kurze Entwicklungszeit	Da die BAM in kurzer Zeit stattfinden wird verbleibt nicht viel Zeit für die Spielentwicklung. Daher muss die Game Engine es ermöglichen in äusserst kurzer Zeit ein komplettes Spiel zu entwickeln.	25
Erlernbarkeit	Aufgrund der kurzen Entwicklungszeit muss der Umgang und das Einsetzen der gewählten Game Engine schnell erlernbar sein.	25
Community	Zur gewählten Game Engine wird eine grosse Community benötigt, da in solchen Fällen viele Fehler bereits bekannt sind und in kurzer Zeit sinnvolle Hilfestellungen gefunden werden können. Auf diese Weise kann das Lösen von Problematiken während der Entwicklung vereinfacht und beschleunigt werden.	20

Tabelle 1: Kriterien inkl. Gewichtung



3.2.2 Skala

Die beiden Varianten werden mit einer Skala zwischen 0 und 5 bewertet. Wenn eine Variante für ein Kriterium die Punktzahl 5 erhält, ist es perfekt dafür geeignet. Erhält eine Variante bei einem Kriterium die Punktzahl 0 so ist es unbrauchbar.

Mit der nachstehenden Formel wird dann im Anschluss zur Punktevergabe die Gesamtpunktzahl der jeweiligen Variante ausgerechnet.

$$\text{GEWICHTUNG} * \text{PUNKTZAHL} = \text{PUNKTE KRITERIUM 1} + \text{PUNKTE KRITERIUM 2} + \dots = \text{TOTAL}$$

Um die Punktzahl der jeweiligen Varianten zu erhalten, muss die definierte Gewichtung mit der gegebenen Punktzahl zum jeweiligen Kriterium multipliziert werden. Die Punktzahlen der einzelnen Kriterien werden anschliessend noch aufsummiert.

3.2.3 Variante 1: PixiJS


Name	Variante 1: PixiJS
Logo	 Abbildung 1: PixiJS Logo
Aktuelle Version	8.10.1
Programmiersprache	JavaScript
Erscheinungsjahr	Juni 2010
Beschreibung	PixiJS ist eine Open-Source Game Engine, womit sich codebasiert Spiele entwickeln lassen. So auch ein 2D Jump and Run. Alle Sprites, Hintergründe oder Bewegungen müssen in JavaScript-Zeilen niedergeschrieben werden. Neben den Verhaltensmustern und Benutzereingaben muss das Platzieren von Sprites immer ausgerechnet werden, was sehr umständlich sein kann. Ein Editor ist nicht existent, was das Platzieren um einiges erleichtern würde.
Vorteile	Nachteile
<ul style="list-style-type: none">- Open-Source- Performance	<ul style="list-style-type: none">- Coding- Längere Entwicklungszeit erwartet

Tabelle 2: Variante 1: PixiJS



3.2.4 Variante 2: Phaser


Name		Variante 1: Phaser
Logo		 Abbildung 2: Phaser Logo
Aktuelle Version		3.9.0
Programmiersprache		TypeScript oder JavaScript
Erscheinungsjahr		April 2013
Beschreibung		
Phaser ist eine Game Engine, mit welcher übers Coding sowie auch über einen Editor ein Spiel entwickelt werden kann. Dabei ist der Editor jedoch Teil von einer nicht freiverfügbaren Version. Somit ist diese Game Engine nur zu Teilen Open-Source. Die relevantesten Features dieser Game Engine, wie der Editor, sind leider kostenpflichtig.		
Vorteile		Nachteile
<ul style="list-style-type: none"> - Mit den meisten Frameworks kombinierbar - Ist ausschliesslich nur für 2D-Spiele geeignet 		<ul style="list-style-type: none"> - Nicht Open-Source - Coding

Tabelle 3: Variante 2: Phaser

3.2.5 Variante 3: GDevelop


Name		Variante 1: GDevelop
Logo		 Abbildung 3: GDevelop Logo
Aktuelle Version		5.5.231
Programmiersprache		Anwendung Editor und Wahl für JavaScript
Erscheinungsjahr		August 2008
Beschreibung		
GDevelop ist eine Game Engine, mit welcher in einem Editor das gesamte Spiel entwickelt werden kann. Es wird also keine Codezeile benötigt, was das Entwickeln eines Spiels äusserst beschleunigt. Wahlweise können dennoch JavaScript-Codeblöcke implementiert werden. GDevelop ist nicht ganz Open-Source, da es unterschiedliche Packages gibt, die teilweise kostenpflichtig sind. Jedoch enthält das freiverfügbare Package von GDevelop alle Features, die für dieses Projekt benötigt werden. Des Weiteren ist die direkte und kostenfreie Veröffentlichung vom Spiel auf gd.games ein grosser Pluspunkt für diese Variante.		
Vorteile		Nachteile
<ul style="list-style-type: none"> - Kostenlose Veröffentlichung auf gd.games - Kurze Entwicklungszeit 		<ul style="list-style-type: none"> - Gewisse Features kostenpflichtig - Debugging etwas erschwert möglich

Tabelle 4: Variante 3: GDevelop



3.2.6 Auswertung

Nun werden die einzelnen Varianten anhand der Kriterien und deren Gewichtung bewertet. Anhand dessen fällt die Entscheidung.

Kriterium	Gewichtung 0-100	Variante 1: PixiJS	Variante 2: Phaser	Variante 3: GDevelop
Open-Source	30	5	1	2
Kurze Entwicklungszeit	25	2	3	5
Erlernbarkeit	25	2	3	5
Community	20	3	3	3
Total Punkt		310	240	<u>370</u>

Tabelle 5: Auswertung

Am Ende des Vergleichs fiel die Wahl recht simpel auf GDevelop. Der grösste Grund dafür ist der kostenlos verfügbare Editor, wodurch die Entwicklungszeit minimiert und der Umgang schnell erlernt werden kann. Ein kostenfreier Editor ist bei keiner der anderen Variante vorhanden.

PixiJS ist zwar Open-Source, jedoch ist mit dieser Variante die benötigte Entwicklungszeit etwas zu gross, um ein komplettes Spiel entwickeln zu können.

Phaser und GDevelop haben beide kostenpflichtige Varianten und eine Open-Source-Variante. Bei Phaser werden dieser Variante jedoch einige Features vorenthalten, welche von grösster Relevanz sind. So beispielsweise den Editor. GDevelop auf der anderen Seite stellt alle Features für die kostenfreie Version zur Verfügung. Die kostenpflichtigen Versionen von GDevelop bieten einige fortgeschrittenere Privilegien.

Dies waren die massgebenden Gründe, die zu einer Wahl von GDevelop führten.

3.3 Umfeld

Entwickelt wird das Spiel anhand der Basis des Variantenentscheids mit der Game Engine «GDevelop». Das Spiel wird anschliessend auf GitLab sowie auch auf GitHub gepusht. Am Ende wird eine Veröffentlichung auf gd.games durchgeführt. Der entstandene Link wird beim Stand an der BAM verbreitet.

3.4 Abgrenzungen

Dieses Projekt grenzt sich von allen anderen im ISCeco bestehenden Anwendungen ab. Es wird ausschliesslich für die BAM entwickelt und dient dazu, dass die Informatiklehre und Informationen über die Ausbildung spielerisch vermittelt werden.

4 Konzept

Nun werden die einzelnen Konzepte für das Jump and Run präsentiert und erläutert.

4.1 Technische Spezifikation

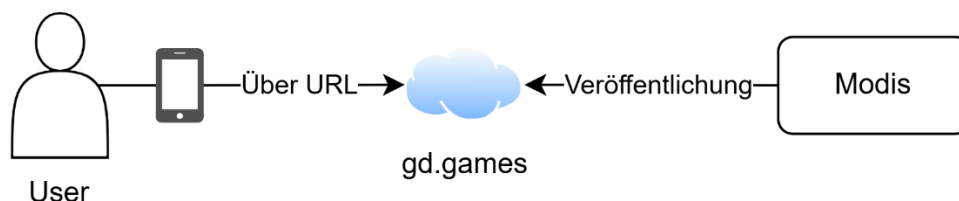





Abbildung 4: Gesamtsystem «Modis»

Das fertiggestellte Spiel soll am Ende in gd.games veröffentlicht werden. «gd.games» ist eine Webapplikation, welche eine Sammlung von Spielen anbietet. All diese Spiele wurden mit GDevelop entwickelt. Durch die Publizierung auf gd.games wird eine URL sowie ein QR-Code erzeugt, worüber direkt das entsprechende Spiel aufgerufen werden kann.

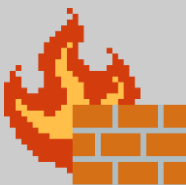
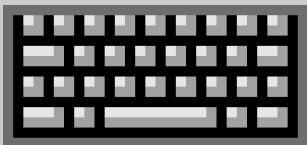

Am Stand der BAM sollen die Interessierten dann diese URL oder QR-Code erhalten und so zunächst nur auf dem Handy das Spiel spielen können. In Zukunft könnte das Game noch um das Spielen auf weiteren Endgeräten erweitert werden, wie beispielsweise auf einem PC.

4.2 Fachspezifikation

Nun werden die einzelnen Objekte, die für das Spielen dieses Spiels von Bedeutung sind, aufgezeigt sowie deren Verhalten beschrieben. So werden Hintergründe und Plattformen nicht oder kaum behandelt. Zur Vollständigkeit werden sie jedoch in kommender Tabelle ebenfalls aufgeführt.

Objektbild	Objektname	Beschreibung
	Super Informatiker	Der Super Informatiker ist die Spielfigur, welches vom User gesteuert wird. Die Figur kann sich bewegen, springen (inkl. Luft- und Wandsprünge), abtauchen sowie Tastaturen einsammeln und werfen. Diese Figur soll alle Module und Üks finden und einsammeln.
	Kabel-Nager	Der Kabel-Nager ist ein sogenannter Feind des Super Informatikers. Der Nager bewegt sich immer wieder von rechts nach links innerhalb eines bestimmten Bereichs. Kollidieren Nager und Informatiker zusammen, so verliert der Super Informatiker ein Herz.
	Wurm	Der Wurm soll auf dieselbe Weise in die Anwendung integriert werden wie der Kabel-Nager.



	Ük-Fragezei- chenbox	<p>Dieses Objekt ist eines, welche der Super Informatiker zu finden hat. Findet er eine Instanz dieses Objekts, so soll der Super Informatiker die Box zerstören, indem er mit der Box kollidiert. Das Kollidieren wird durch das Springen oder Abtauchen der Spielfigur erreicht.</p> <p>Darüber hinaus soll dem User nach dem Entfernen eine Frage inkl. Antwortmöglichkeiten gestellt werden. Beantwortet er diese richtig, so erhält er ein Herz mehr dazu. Beantwortet der User die Frage falsch, so verliert er ein Herz.</p>
	Firewall	<p>Bei der Firewall handelt es sich um einen weiteren Feind. An der Wand, an der der Spieler vorbeimuss, entfacht in regelmässigen Abständen ein Feuer. Wird der Benutzer vom Feuer getroffen, so verliert er ein Herz.</p>
	Modul-Frage- zeichenbox	<p>Die Modul-Fragezeichenbox verhält sich gleich wie die Box des Üks. Der einzige Unterschied liegt darin, dass der Spielfigur bei einer korrekten Antwort auf die gestellte Frage einige Tastaturen zum Einsammeln gegeben werden.</p>
	Tastatur	<p>Tastaturen liegen in der Map verteilt und können vom User eingesammelt werden. Die eingesammelten Tastaturen kann der Benutzer dann auch werfen. Wird ein Gegner mit einer solchen Tastatur getroffen, so wird der Gegner zerstört. Trifft die Tastatur jedoch zuerst auf einen anderen Block oder Objekt, so wird nur die Tastatur zerstört.</p>
	Wall	<p>Das Wall-Objekt ist jenes, welches die Map links, rechts und and der Decke mit einer Wand schliesst, so dass der User nicht aus der Map fallen kann.</p>
	Gras	<p>Das Grasobjekt wird als Plattform genutzt. So soll sich der Benutzer auf diesem Objekt bewegen können.</p>

4.3 Use Case

Für einen guten Spielfluss werden einige Funktionalitäten für den Benutzer des Spiels benötigt. Diese Fälle werden mit der kommenden Abbildung eines Use Case-Diagramms und den dazugehörigen Use Case-Beschreibungen dargestellt und erläutert.

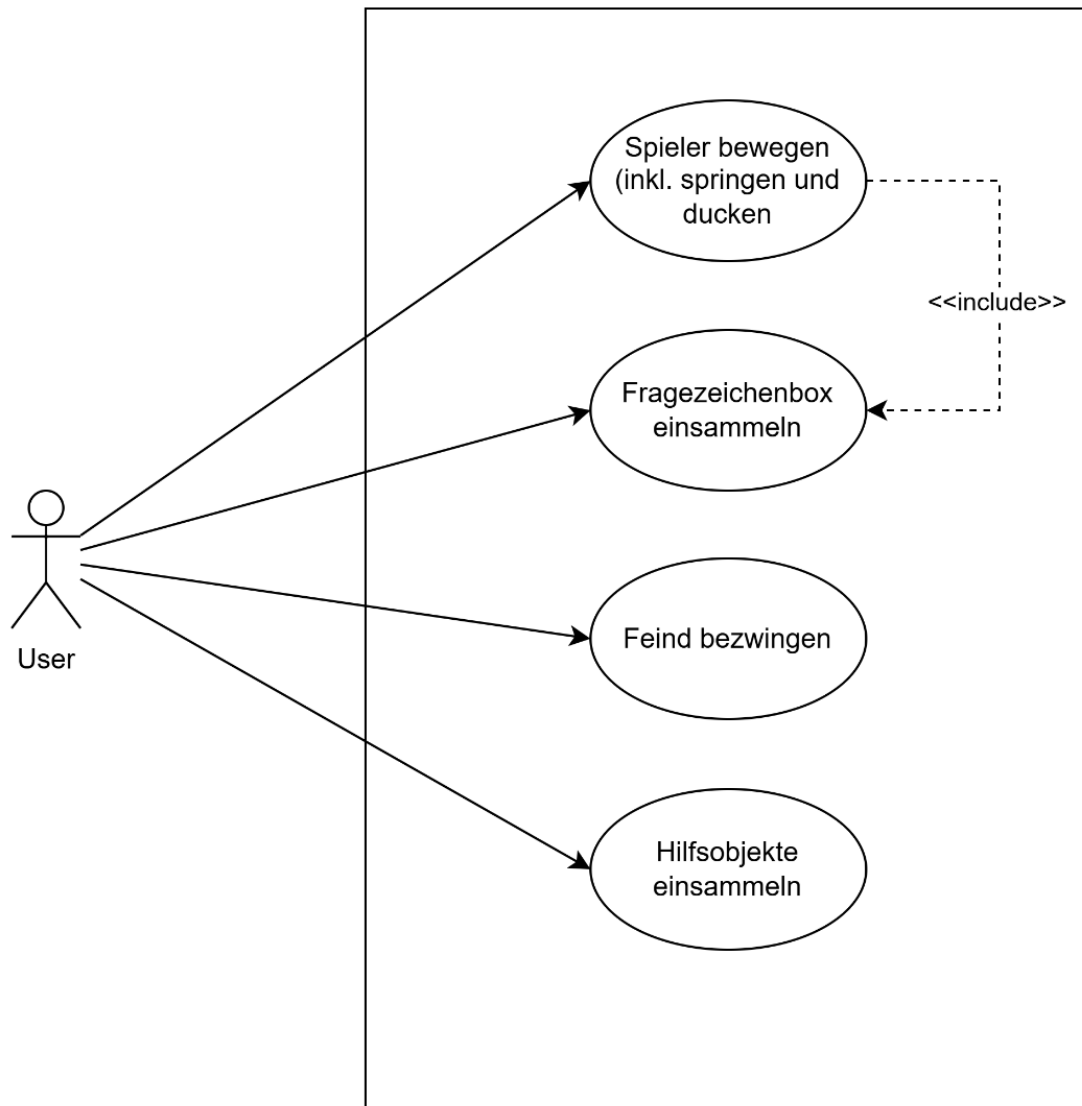


Abbildung 6: Use Case Diagramm



Die einzelnen Beschreibungen folgen in den vereinzelt Tabellen, die folgend abgebildet werden.

Use Case Eigenschaften	Beschreibung
Name	Spieler bewegen (inkl. springen und ducken)
Kurzbeschreibung	Der Spieler sollte seinen Spieler-Avatar nach links und rechts bewegen, sowie springen und ducken können.
Beteiligte	Player-Objekt
Auslöser	Der Benutzer möchte die Spielfigur bewegen
Vorbedingung	Spiel geöffnet und Run gestartet
Normalablauf	<ol style="list-style-type: none">1. Der Benutzer drückt auf den Pfeil, der nach links zeigt<ol style="list-style-type: none">a. Die Spielfigur läuft nach links2. Der Benutzer drückt auf den Pfeil, der nach rechts zeigt<ol style="list-style-type: none">a. Die Spielfigur läuft nach rechts3. Der Benutzer drückt auf den Pfeil, der nach oben zeigt<ol style="list-style-type: none">a. Die Spielfigur führt einen Sprung aus4. Der Benutzer drückt auf den Pfeil, der nach unten zeigt<ol style="list-style-type: none">a. Die Spielfigur duckt sich und fällt so schneller auf den Boden
Ablauf mit Fehlern	-
Ergebnis	Die Spielerfigur bewegt sich nach rechts, links, springt oder duckt abhängig vom Pfeil, der bei den Steuerungstasten gedrückt wird.
Nachbedingung	-
Hinweise	<ul style="list-style-type: none">- Der Spieler kann sich an Wänden festhalten. Auf diese Weise fällt er langsamer- Der Spieler kann von Wandkanten abspringen- Der Spieler kann in der Luft ein weiteres Mal springen

Tabelle 6: Use Case «Spieler bewegen»



Use Case Eigenschaften	Beschreibung
Name	Fragezeichenbox einsammeln
Kurzbeschreibung	Der Benutzer soll Üks und Module in Form von Fragezeichenboxen finden und einsammeln, indem er das Objekt zerstört. Das Zerstören wird durch das Springen und Ducken erfolgen.
Beteiligte	Spieler, Fragezeichenboxen
Auslöser	Spieler findet eine Fragezeichenbox
Vorbedingung	Spieler findet eine Fragezeichenbox; Das Spiel läuft und ein Run wurde gestartet
Normalablauf	<ol style="list-style-type: none">1. Der Spieler springt oder duckt sich, um gezielt das entsprechende Modul/Ük zu zerstören. Dies geschieht, indem die Spielfigur mit der entsprechenden Fragezeichenbox kollidiert.2. Es erscheint eine neue Benutzeroberfläche, bei der eine Frage gestellt wird und vier Antwortmöglichkeiten angezeigt werden. Der Benutzer soll die richtige Antwortmöglichkeit auswählen.3. Der Benutzer klickt die richtige Antwortmöglichkeit an. Dies wird ihm so angezeigt. Handelt es sich beim eingesammelten Objekt um ein Modul, so erhält er ein paar Tastaturen zum Werfen dazu. Handelt es sich um einen Ük, so erhält er ein Leben dazu
Ablauf mit Fehlern	<ol style="list-style-type: none">1. Wählt der Benutzer die falsche Antwortmöglichkeit aus, so verliert er ein Leben
Ergebnis	Der Benutzer sammelt ein Modul oder ein Ük ein und bekommt Informationen über die Informatiklehre vermittelt.
Nachbedingung	-
Hinweise	-

Tabelle 7: Use Case «Fragezeichenbox einsammeln»



Use Case Eigenschaften	Beschreibung
Name	Feind bezwingen
Kurzbeschreibung	Der Benutzer soll Feinde bezwingen können.
Beteiligte	Player, Enemies, Tastatur
Auslöser	Der Spieler steht vor einem Feind
Vorbedingung	Die Spielfigur ist in Besitz von bereits eingesammelten Tastaturen. Das Spiel läuft und ein Run wurde gestartet
Normalablauf	<ol style="list-style-type: none">1. Der Spieler steht vor einem Feind2. Der Spieler wirft eine Tastatur auf den Gegner3. Sobald die Tastatur mit dem Gegner kollidiert, wurde er zerstört.
Ablauf mit Fehlern	<ol style="list-style-type: none">1. Eine Firewall ist nicht zerstörbar2. Sollte eine Zerstörung vom Gegner schiefgehen und die Spielfigur kollidiert mit dem Feind, so verliert der Benutzer ein Leben. Bei der Firewall muss der Spieler mit dem Feuer kollidieren.3. Der Bug kann auch zerstört werden indem auf ihn geduckt wird.4. Wird eine Tastatur geworfen, die anschliessend mit einem anderen Objekt kollidiert, so wird nur die Tastatur zerstört
Ergebnis	Der Feind wurde zerstört und der Spieler kann das Jump and Run einfacher fortfahren
Nachbedingung	-
Hinweise	<ul style="list-style-type: none">- Hat der Benutzer keine Tastaturen mehr, so kann er keine mehr werfen

Tabelle 8: Use Case «Feind bezwingen»



Use Case Eigenschaften	Beschreibung
Name	Hilfsobjekte einsammeln
Kurzbeschreibung	Die Spielfigur soll Items wie beispielsweise Tastaturen einsammeln können: Diese helfen ihm im weiteren Verlauf des Spiels.
Beteiligte	Spieler, Tastatur
Auslöser	Spieler steht vor einer Tastatur
Vorbedingung	Das Spiel läuft und ein Run wurde gestartet
Normalablauf	<ol style="list-style-type: none">1. Der Spieler bewegt sich auf die Tastatur zu bis sie miteinander kollidieren2. Die Tastatur wird eingesammelt und kann verwendet werden. In der Map vom Run ist die Tastatur an der Stelle nicht mehr aufzufinden.
Ablauf mit Fehlern	-
Ergebnis	Der Spieler hat eine Tastatur eingesammelt und kann diese nun verwenden
Nachbedingung	-
Hinweise	<ul style="list-style-type: none">- Zukünftig kann dieser Use Case mit weiteren beliebigen Objekten angewandt werden. Die Tastatur bleibt allerdings zunächst das einzige Hilfsobjekt, weshalb von der Tastatur im Normalablauf die Rede ist.

Tabelle 9: Use Case «Hilfsobjekte einsammeln»



4.4 Benutzeroberflächen

Insgesamt sollen für beide Fachrichtungen der Informatik (Applikationsentwicklung und Plattformentwicklung) jeweils für jedes Lehrjahr in der Informatiklehre ein Level entstehen (Pro Fachrichtung 4 Level). Jedes Level beinhaltet dabei einen Run. Darüber hinaus wird es einen Endboss geben, welcher die IPA symbolisiert. Hierbei soll dann ein Quiz entstehen.

Wie die einzelnen Runs aufgebaut und strukturiert werden, wird einem grundsätzlich frei überlassen. Es ist jedoch darauf zu achten, dass sich die Schwierigkeit von Level zu Level immer mehr steigert.

5 Realisierung

Mit den kommenden Unterkapiteln soll die Entwicklung mit GDevelop kurz aufgezeigt und erläutert werden.

5.1 Verweisung auf Beispielprojekt

Als Referenz für das endgültige Spiel gilt der Prototyp «modis», welcher in GitLab und GitHub aufzufinden ist.

5.1.1 Genutzte Extensions

Für den Prototypen wurden einige Extensions verwendet. Daher werden in einer kommenden Tabelle kurz die einzelnen Extensions aufgeführt und erläutert.

Extension	Erläuterung
AdvancedJump	Mit dieser Extension werden Sprünge aber gerade Air Jumps und Wandsprünge ermöglicht. Dazu muss nach dem Hinzufügen der Extension beim entsprechenden Objekt das entsprechende Verhalten hinzugefügt werden.
AdvancedProjectile	Durch diese Extension lassen sich Objekte, sobald eine Instanz davon in der Szene enthalten ist, in eine bestimmte Richtung bewegen. Dies geschieht ohne weitere Aktionen. Nach dem Hinzufügen der Extension musste das entsprechende Verhalten beim entsprechenden Objekt hinzugefügt werden.
MouseHelper	Diese Extension ermöglicht es ein beliebiges Objekt als Cursor verwenden zu können. Sobald die Extension hinzugefügt wurde, muss das entsprechende Verhalten beim entsprechenden Objekt hinzugefügt werden.

Tabelle 10: Extensions

5.2 Entwicklung mit GDevelop

Nun wird erläutert, wie die Entwicklung mit der GDevelop-IDE vonstattengeht. Es werden die wichtigsten Aspekte bei der Arbeit mit besagter Game Engine beschrieben.

5.2.1 Variablen

Es gibt in GDevelop drei unterschiedliche Arten von Variablen, die die Sichtbarkeit dieser beeinflussen. So gibt es globale und Szenenvariablen. Die globalen Variablen sind über das ganze Projekt erreichbar, während die Szenenvariablen rein in einer Szene sichtbar und ansprechbar sind.

Neben diesen Variablentypen existieren noch Spritevariablen. Dies sind Variablen, die einem gewissen Sprite / Objekt zugewiesen wurden. Sie sind somit nur über entsprechendes Objekt auch erreichbar.



5.2.2 Event-System

Um Aktionen durchführen lassen zu können wird das Event-System von GDevelop verwendet. Dabei werden Bedingungen eingesetzt, die erfüllt werden müssen, damit gewisse Aktionen durchgeführt werden können. Wie bei den herkömmlichen Events wird gehorcht, bis die Bedingungen erfüllt werden. Bei Erfüllung der Bedingungen werden im Anschluss die entsprechenden Aktionen nacheinander durchgeführt.

5.2.3 Behaviors und Animations

Bei der Erstellung von neuen Objekten wird nicht nur ihr Aussehen festgelegt, sondern auch ihr Verhalten. Die Behaviors, die für ein Objekt festgelegt werden, bestimmen wie sich die Instanzen des entsprechenden Objekts grundsätzlich verhalten. Die Behaviors können dann noch selbst konfiguriert werden. Wenn beispielsweise eine Instanz eines Gras-Objekts bei der Spielszene eingefügt wird, so wird dieser erst als Boden oder Plattform angesehen, wenn das entsprechende Verhalten zum Objekt hinzugefügt wurde.

Neben den Verhalten gibt es dann noch die Animationen. Jedes Objekt kann eine beliebige Anzahl an Animationen haben. Die Animationen bestehen aus einer angereihten Abfolge von Bildern, die innerhalb einer gewissen Zeit durchgeführt werden.



6 Abbildungsverzeichnis

Abbildung 1: PixiJS Logo	4
Abbildung 2: Phaser Logo	5
Abbildung 3: GDevelop Logo	5
Abbildung 4: Gesamtsystem «Modis»	7
Abbildung 5: Spielobjekte	9
Abbildung 6: Use Case Diagramm	10

7 Tabellenverzeichnis

Tabelle 1: Kriterien inkl. Gewichtung	3
Tabelle 2: Variante 1: PixiJS	4
Tabelle 3: Variante 2: Phaser	5
Tabelle 4: Variante 3: GDevelop	5
Tabelle 5: Auswertung	6
Tabelle 6: Use Case «Spieler bewegen»	11
Tabelle 7: Use Case «Fragezeichenbox einsammeln»	12
Tabelle 8: Use Case «Feind bezwingen»	13
Tabelle 9: Use Case «Hilfsobjekte einsammeln»	14
Tabelle 10: Extensions	15
Tabelle 11: Quellenverzeichnis	17
Tabelle 12: Glossar	18

8 Quellenverzeichnis

Quellenthematik	Quelle	Stand (Zuletzt besucht)
GDevelop	GDevelop: Kostenlos, schnell, einfache Spiele-Engine	07.07.2025
Phaser	Phaser - A fast and fun HTML5 game framework	07.07.2025
PixiJS	PixiJS	07.07.2025
Game Engine Definition	Game-Engine Definition und Beschreibung	07.07.2025
BAM Definition	https://bam.ch/de	07.07.2025

Tabelle 11: Quellenverzeichnis



9 Glossar

Abkürzung	Begriff	Erläuterung
IDE	Integrierte Entwicklungsumgebung	Integrierte Entwicklungsumgebungen sind Programme, welche zur vereinfachten Entwicklung von Anwendungen verwendet werden.
BAM	Berufs- und Ausbildungsmesse Bern	Dies ist eine Veranstaltung zur Vermittlung von Informationen über die unterschiedlichen Bildungswege und Berufe.
	Szene	Eine Szene ist das Pendant zu einer View. Es ist eine Ansicht, die der Spieler am Ende zu Gesicht bekommen soll und ist ein Teil des Spiels. Ein Spiel kann schnell mehrere solcher Szenen benötigen.
	Sprite	Ein Sprite beschreibt ein grafisches Objekt, welches in der Spielentwicklung verwendet wird und eine gewisse Rolle einnimmt. So handelt es sich beispielsweise beim Super Informatiker oder Kabel-Nager um einen Sprite.
	GDevelop	GDevelop ist eine Game Engine, welche unter anderem eine eigene IDE bereitstellt.
	gd.games	gd.games ist eine Webapplikation, auf der alle möglichen Spiele, die mit GDevelop entwickelt wurden, veröffentlicht und gespielt werden können.
	Game Engine	Eine Game Engine ist eine Software-Plattform, welche zur Spielentwicklung verwendet werden.
	Feind (engl. Enemy)	Ein Feind (oder Enemy) ist ein Sprite, welches das Ziel hat den Super Informatiker (gesteuert vom Benutzer) zu zerstören.
	Behaviors	Ein Verhalten ist sozusagen eine bereits vorgefertigte Funktion, die einem Sprite hinzugefügt werden kann. Dieses Objekt erhält dann gewisse Eigenschaften oder Funktionen. So kann beispielsweise dem Super Informatiker ein Behavior namens «Platformer Object» hinzugefügt werden, wodurch er sich bewegen, springen und fallen lässt.

Tabelle 12: Glossar