**Dokumentation**

**des Softwareprojektes**

mit der Software Unity



### Bearbeitungszeitraum: 27.01.2022-31.03.2023

### Name des Faches: Informatik

### Verfasser: Peter Wolter, Owen Liedtke

### Lehrkraft: Herr Schütze

Inhaltsverzeichnis

[1 Programmierer Dokumentation 3](#_Toc131192286)

[1.1 Allgemeines 3](#_Toc131192287)

[1.2 Pushback-Fähigkeit (pushbackfigure) 3](#_Toc131192288)

[1.3 Spielerfähigkeiten ausführen (startability und useability) 3](#_Toc131192289)

[1.4 Feldeingabe für Fähigkeiten prüfen (tilevalidforability) 4](#_Toc131192290)

[1.5 Figureninformation ausgeben (getinfotext) 4](#_Toc131192291)

[1.6 InputSystem 4](#_Toc131192292)

[2 Methodendokumentation 5](#_Toc131192293)

[2.1 Einstellung der Bilddateien 5](#_Toc131192294)

[2.2 Verwendung von Tags 5](#_Toc131192295)

[2.3 StartCoroutine und IEnumerator 5](#_Toc131192296)

[3 Benutzerdokumentation 6](#_Toc131192297)

[4 Datendokumentation 6](#_Toc131192298)

[4.1 Levelklasse (LevelProperties) 6](#_Toc131192299)

[4.2 Fähigkeitsaufruf 7](#_Toc131192300)

[4.3 Spielergebnisspeicherung (GameResult-Skript) 8](#_Toc131192301)

[5 Entwicklungsdokumentation 9](#_Toc131192302)

[Literaturverzeichnis 9](#_Toc131192303)

[Plagiatszettel 11](#_Toc131192304)

# Programmierer Dokumentation

## Allgemeines

Die Programmierung der Software erfolgte in C# und ist in 23 Skripte eingeteilt. Ein Skript (Name SwitchLevel) ist für den Übergang der einzelnen Szenen verantwortlich und ein weiteres (GameResult) für die Verkündung des Spielergebnisses, alle anderen Skripte konzentrieren sich explizit auf die Spielmechaniken.

Die Skripte „PlayerInfo“, „RoundDisplay“, „EnergyDisplay“ und „MovesLeftDisplay“ sorgen für die Aktualisierung der UI-Texte, wie Energieanzahl, Objektinformation oder die aktuell auszuführende Aktion, etc. in der Szene.

## Pushback-Fähigkeit (pushbackfigure)

Pushback - sind die beiden Feldpunkte (P = Ursprungspunkt, S = Zielpunkt)

Die Funktion pushback verschiebt die Figur von S in Richtung des Differenzenvektors . Allerdings nur um je maximal 1 Einheit in jede Richtung, also , wobei Math.Sign das Vorzeichen der Differenz angibt – also +1,-1 oder 0.

Somit ist die neue Position P‘:

Es ergibt sich

new int[] {x+Math.Sign(x-fromx), y+Math.Sign(y-fromy)}

als Array der neuen Position.

## Spielerfähigkeiten ausführen (startability und useability)

Jegliche Verwendung einer Spielerfähigkeit wird in zwei Funktionen im FigureActions Skript eingeteilt: startability und useability.

Erfolgt die Betätigung eines Angriffs-Buttons (siehe ButtonActionScript), so wird zunächst die startability Funktion ausgeführt, welche den Index der zu benutzenden Fähigkeit in selectedability zwischenspeichert. Erfolgt nun eine weitere Eingabe in Form eines Feld-, Figuren- oder Beweguns-Buttons, so wird die Funktion useability aufgerufen. Es wird geprüft, ob die Eingabe für die Fähigkeit Sinn ergibt bzw. valide ist (Siehe Funktion tilevalidforability), und daraufhin erfolgt die wirkliche Ausführung der vorher abgespeicherten Fähigkeit in der Funktion useability.

## Feldeingabe für Fähigkeiten prüfen (tilevalidforability)

Obwohl es 9 verschiedene Fähigkeiten gibt, benötigen diese nur 4 verschiedene Bedingungen, um zu prüfen, ob das Eingabefeld valide ist. Für diese Einteilung wird die Variable k verwendet und ist in einem Array eingetragen (bei dem das Element an der Stelle des Index‘ der Fähigkeit gleich dem nötigen k Wert für diese Fähigkeit ist).

Die erste Bedingung (k=0) für das Feld ist die ohne Bedingung bzw. ohne nötige Eingabe (Fähigkeit 2).

Ein Bild, das Diagramm enthält.

Automatisch generierte BeschreibungDie zweite Bedingung (k=1) für das Feld, ist die, bei dem das Feld genau 1 Feldabstand zur Figur haben darf (Siehe Abb. 1) (Fähigkeiten 1, 3, 6). Dabei muss entweder der Abstand in x-Richtung:

genau 1 sein, oder der Abstand in y-Richtung:

genau 1 sein. In beiden Fällen ist deren Summe also genau 1:

Abbildung 1

Folgich ist das die zu erfüllende Bedingung für das Eingabefeld.

Die dritte Bedingung (k=2) für das Feld, ist die, bei dem das Feld sich in der gleichen Reihe befinden muss (Fähigkeit 4). Dafür muss der Abstand in x-Richtung oder der Abstand in y-Richtung null sein (Es kann auf die Betragsbildung verzichtet werden, da der Wert 0 annehmen muss).

Bei der vierten Bedingung (k=3) darf das Feld nur einen festgelegten maximalen Abstand n zum Feld haben (Abstand eines diagonalen Feldes ist 2). Daraus folgt, dass die Summe der Abstände in beide Richtungen kleiner gleich n sein musss:

## Figureninformation ausgeben (getinfotext)

Die Funktion gibt die Figureninformationen als string aus, wobei zwischen Spieler und Gegner unterschieden wird.

Es wird mehrfach folgende Struktur verwendet:

string s = player?"Player":"Enemy";

Dabei bedeutet der Code lediglich: wenn boolean player wahr ist, ist s gleich “Player“, sonst „Enemy“. Mit dieser Struktur wird es bedeutend einfacher, die Ausgabe des strings s zu verändern, je nach dem, ob es sich um einen Spieler handelt oder um einen Gegner (Siehe Handbuch/User Interface).

## InputSystem

Jeder Input in Form eine Klicks auf ein Feld, eine Figur, eine Movement-Button wird von den dezentralen Skripten (ClickDetector, ActionButtonScript) in einem zentralen Skript InputSystem zusammengefasst. Wird eine Figur geklickt, wird dies aus den genannten dezentralen Skripten dem zentralen Skript gemeldet und die Funktion figureclickdetected aufgerufen. Wird ein Feld oder ein Movement-Button gedrückt, so wird logischerweise eines der beiden Skripte fieldclickdetected oder buttonclickdetected. Dadurch muss die der Code, der nach einem Klick ausgeführt werden soll, nur einmal aufgeschrieben werden anstelle von mehreren Malen in den verschiedenen Skripten.

# Methodendokumentation

## Einstellung der Bilddateien

Damit alle verwendeten Bilder (dazu zählen Feld-Designs, Figuren-Designs, etc.) von der Größe her einheitlich sind, verwendeten wir immer 128x128 Pixel Dateien. Außerdem stellten wir das Attribut Filtermode im Inspektor der Bilddateien auf Point eingestellt, damit

## Verwendung von Tags

Zur Hilfestellung in der Programmierung benutzten wir Tags bei den GameObjects, um sie in den Skripten besser ansprechen zu können. Alle GameObjects, die eine Figur repräsentieren sollten, besaßen den Tag „Figure“; alle, die ein Feld repräsentieren sollten, den Tag „Grid“; alle

GameObjects, die Gegner Announces (Ausrufezeichen) repräsentierten, den Tag „Exclamation“. Das machte es sehr viel einfacher, sich auf die GameObjects im Code zu beziehen.

## StartCoroutine und IEnumerator

Bei der Darstellung fast aller Skripte, des Gegnerzuges verwendeten wir die Funktion StartCoroutine, welche es uns ermöglichte, das System eine bestimmte Zeit warten zu lassen, damit die Aktionen der Gegner für den Spieler besser nachvollziehbar werden. Dabei musste man lediglich die Zeile:

public void test(){

    StartCoroutine(waited());

}

IEnumerator waited(){

    //Skript vorher

yield return  new WaitForSeconds(2);

    //Skript nachher

}

in einer IEnumerator Funktion, schreiben. Zwecks der Übersichtlichkeit beginnen im Code alle Namen der IEnumerator Funktionen mit waited.

# Benutzerdokumentation

Um das Programm zu starten. Muss lediglich der Ordner des Spielsgeöffnet werden und die sich darin befindende .exe-Datei geöffnet werden. Diese ist mit dem Namen softwareprojekt gekennzeichnet.

Die Verwendung des Programmes und die Erklärung des Spiels wird in userem Handbuch beschrieben.

# Datendokumentation

## Levelklasse (LevelProperties)

Das Programm speichert die Leveldaten in einer Level Klasse (Skriptname LevelProperties). Dabei steht das Array playerablilties für die Zuordnung der Standardattacke und Spezialattacke zu den jeweiligen Spielern (1. Element gibt die Attacken des 1. Spielers an, etc.).

Im Skript TurnManager existiert außerdem noch die Variable step, welche für den Status im aktuellen Spieler

## Funktionen setposition und gridvalue

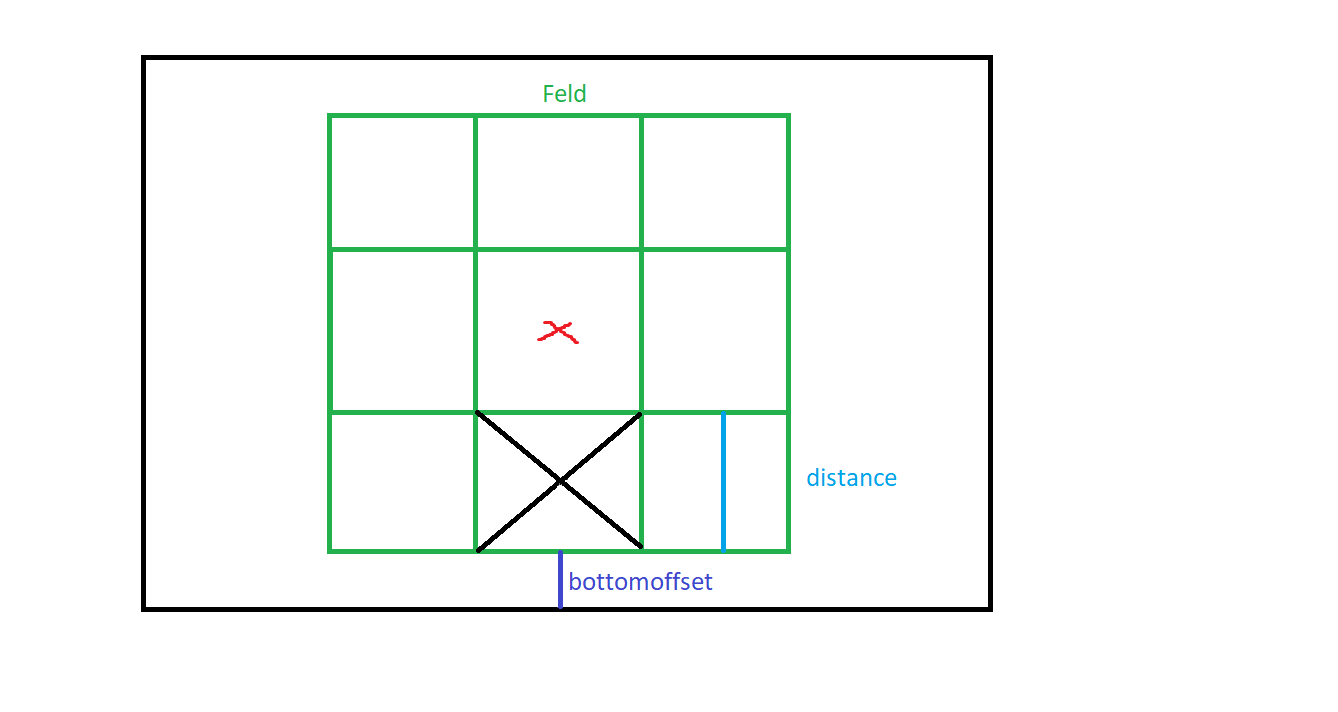
Im FigureActions Skript befinden sich die Funktionen setposition und gridvalue, welche die Weltkoordinaten der GameObjects in die Koordinaten im 9x9 Feld umwandeln. Dafür mussten wir zunächst die Lage des Feldes auf dem Screen mit den Variablen distance und bottomoffset festlegen. Hierbei platzierten wir die Feldmitte genau in die Bildschirmmitte (Siehe Abb. 2). Durch Verändern des Attributes Pixels per Unit im Inspektor der Bild-Dateien der Felder ermittelten wir die Feldgröße, bei der genau 9 Felder vertikal in den Sichtbereich der Kamera hinein passen. Es ergab sich ein angemessen großes 9x9 Feld bei 1.15 Pixels per Unit. Daraufhin fügten wir in den GameObjects der Felder die Komponente Box Collider 2D hinzu, mithilfe dessen wir unter dem Reiter Edit Collider die Größe des anklickbaren Bereiches des Feldes festlegen konnten, was exakt der visuellen Größe des Feldes am Bildschirm entsprach. Deshalb waren die Breite und Länge des Box Colliders (Attribut Size) und die Breite und Länge des Feldes gleich groß, womit wir den Wert für distance von ca. 110 Units herausbekamen (wir rundeten ihn der Einfachheit halber bewusst ab).

Abbildung 2 – Projekt Skizze

Es ergab sich also folgende Gleichung für die Berechnung der Bildschirmkoordinaten:

(Für die x-Koordinate, mit x=x-te Spalte von links)

Camera.main.pixelWidth/2f+distance\*(x-4)

(Für die y-Koordinate, mit x=x-te Reihe von unten)

bottomoffset+distance/2f+distance\*y

Damit die Felder keinen Abstand in z-Richtung von der Kamera haben und somit auf dem Bildschirm zu sehen sind, musste der z-Wert auf 100 gestellt werden, da der Canvas ebenfalls eine Plane Distance (Canvas-Komponente) von 100 besaß.

Zusätzlich war eine Skalierung in x- und y-Richtung für verschiedenartige Bildschirme notwendig, welche sich aus dem Quotient der Pixelanzahl des Bildschirms und der Kamera errechnen ließ, um die Bildschirmpositionen an verschiedene Bildschirmarten anzupassen.

Als letztes mussten die errechneten Screen-Koordinaten in Weltkoordinaten mit der von Unity gestellten Funktion ScreenToWorldPoint() umgerechnet werden, wobei diese von der Kamera der Szene ausgeführt werden musste.

Daraus entwickelte sich folgende Codezeile:

tile.transform.position = Camera.main.ScreenToWorldPoint(new Vector3 (((float)Camera.main.pixelWidth/2f+distance\*(x-4))\*ws, (bottomoffset+distance/2f+distance\*y)\*hs, 100f));

Die Funktion gridvalue ist nichts weiter als die Umkehrfunktion der Funktion setposition. Dafür mussten zunächst einmal die Weltkoordinaten in Screenkoordinaten umgerechnet werden:

Vector3 newpos = Camera.main.WorldToScreenPoint(pos);

Jetzt konnte man die für setposition herausgefundene Rechnung umkehren und nach x bzw. y umstellen:

Erste Umstellung nach x, wobei v das Ergebnis der Funktion setposition ist:

v=(((float)Camera.main.pixelWidth/2f+distance\*(x-4))\*ws |:ws

v/ws=Camera.main.pixelWidth/(2f\*ws)+distance\*(x-4) |-...Width/(2f\*ws)

v/ws-Camera.main.pixelWidth/(2f\*ws)=distance\*(x-4) |:distance

(v/ws-Camera.main.pixelWidth/(2f\*ws))/distance=x-4 |+4

(v/ws-Camera.main.pixelWidth/(2f\*ws))/distance+4=x

Hängt man nun noch (float) an pixelWidth heran und bildet einen Integer-Wert mit (int), so erhält man folgende Formel:

gridc[0]=(int)((newpos.x/gridmanager.ws-(float)Camera.main.pixelWidth/2f)/gridmanager.distance-100)+104;

(die -100 und +100 sind dafür, dass (int)(float) die Zahl immer aufrundet, und nicht beim Vorzeichenwechsel auf einmal abrundet:

(int)(-0.5)=0 Aufrunden

(int)(0.5)=0 Abrunden

Das selbe Prinzip lässt sich auf die y-Koordinate anwenden:

v=(bottomoffset+distance/2f+distance\*y)\*hs |:hs

v/hs=bottomoffset+distance/2f+distance\*y |-bottomoffset

v/hs-bottomoffset=distance/2f+distance\*y |-distance/2f

v/hs-bottomoffset-distance/2f=distance\*y |:distance

(v/hs-bottomoffset-distance/2f)/distance=y

y=(v/hs-bottomoffset)/distance-1/2

gridc[1]=(int)(newpos.y/(gridmanager.distance\*gridmanager.hs)-gridmanager.bottomoffset/gridmanager.distance-1/2);

## Fähigkeitsaufruf

Zur Speicherung der Daten, welche Figur welche Fähigkeiten ausführen kann, hinterlegten wir in der FigureAttributeScript – Klasse (Klasse für die Figur) den Index der Fähigkeit. Die folgende Tabelle soll dies veranschaulichen:

|  |  |  |
| --- | --- | --- |
| Index | Fähigkeitsname | Fähigkeitsbeschreibung |
| 1 | Knüppelschlag | Schaden auf dem Feld links, rechts, oben oder unten von der Figur mit Verschiebung des Gegners um ein Feld nach hinten |
| 2 | Blitz | Schaden an allen Gegnern |
| 3 | Schlag | Schaden auf dem Feld links, rechts, oben oder unten von der Figur |
| 4 | Bombenwurf | Feld in derselben Reihe; alle umliegenden Felder erhalten Schaden und Figuren werden zurückgedrückt |
| 5 | Speerwurf | Feld in derselben Reihe; Gegner erhält Schaden, der sich pro Feldabstand erhöht und wird verschoben um ein Feld nach hinten |
| 6 | Sense | Schaden auf den drei Feldern links, rechts, oben oder unten von der Figur im Radius von 1 (Mehr Schaden, je mehr Figuren angegriffen werden) |
| 7 | Fesselung | Deaktiviert die Attacke, die ein Gegner im nächsten Zug plant |
| 8 | Bogenschuss | Schaden auf einem Feld mit max. 2 Abstand zu der Figur (Diagonal zählt als Abstand von 2) |
| 9 | Windstoß | Verschiebung der nächsten Figur in allen deinen Reihen bis zu einem Gebirge, Haupthaus, oder zum Feldrand. Dabei wird die Figur geschädigt und |

Je nach Fähigkeitstyp und der damit einhergehenden Bedingung für die Feldeingabe (Feld, an dem die Attacke ausgeführt werden soll) haben wir die switch-case-Klausel benutzt. Dadurch sind die Grenzwerte für den Index: [1,9] (mindestens 1, maximal 9)

## Spielergebnisspeicherung (GameResult-Skript)

Nachdem ein Spiel gewonnen oder verloren wurde, wird das Ergebnis in dem Array won abgespeichert. Dabei steht das 1. Element vom Array dafür, ob das 1. Level gewonnen oder noch nicht gewonnen wurde, und das 2. Element dafür, ob das 2. Level gewonnen oder noch nicht gewonnen wurde. In dem Array wonstatus wird zusätzlich für jedes Level abgespeichert, mit wie vielen noch stehenden Häusern der Spieler gewonnen hat (daran wird das beste Spiel gemessen). Sein bestes Ergebnis wird vermerkt und in der LevelMenu Szene angezeigt.

# Entwicklungsdokumentation

Aufgrund einiger Problem mit dem Projekt, mussten wir unser Github repository mehrfach neu erstellen. Deshalb steht die kann man die Fortschritte unseres Projekts vom kompletten Anfang an in dem repository Informatik\_Projekt sehen und danach auf dem software\_projekt repository. Die letzten Änderungen (ab dem 17. März des Projektes wurden ebenfalls nicht mehr auf der GitHistory aktualisiert und werden im Folgenden dokumentiert:

Farben im Skript ändern:

https://answers.unity.com/questions/1144913/how-to-getcomponent-image-coloralpha.html 27.03.2023

Veränderung 3 Spieler-Reiter mit Farben

Koordinatensystem in Unity verstehen

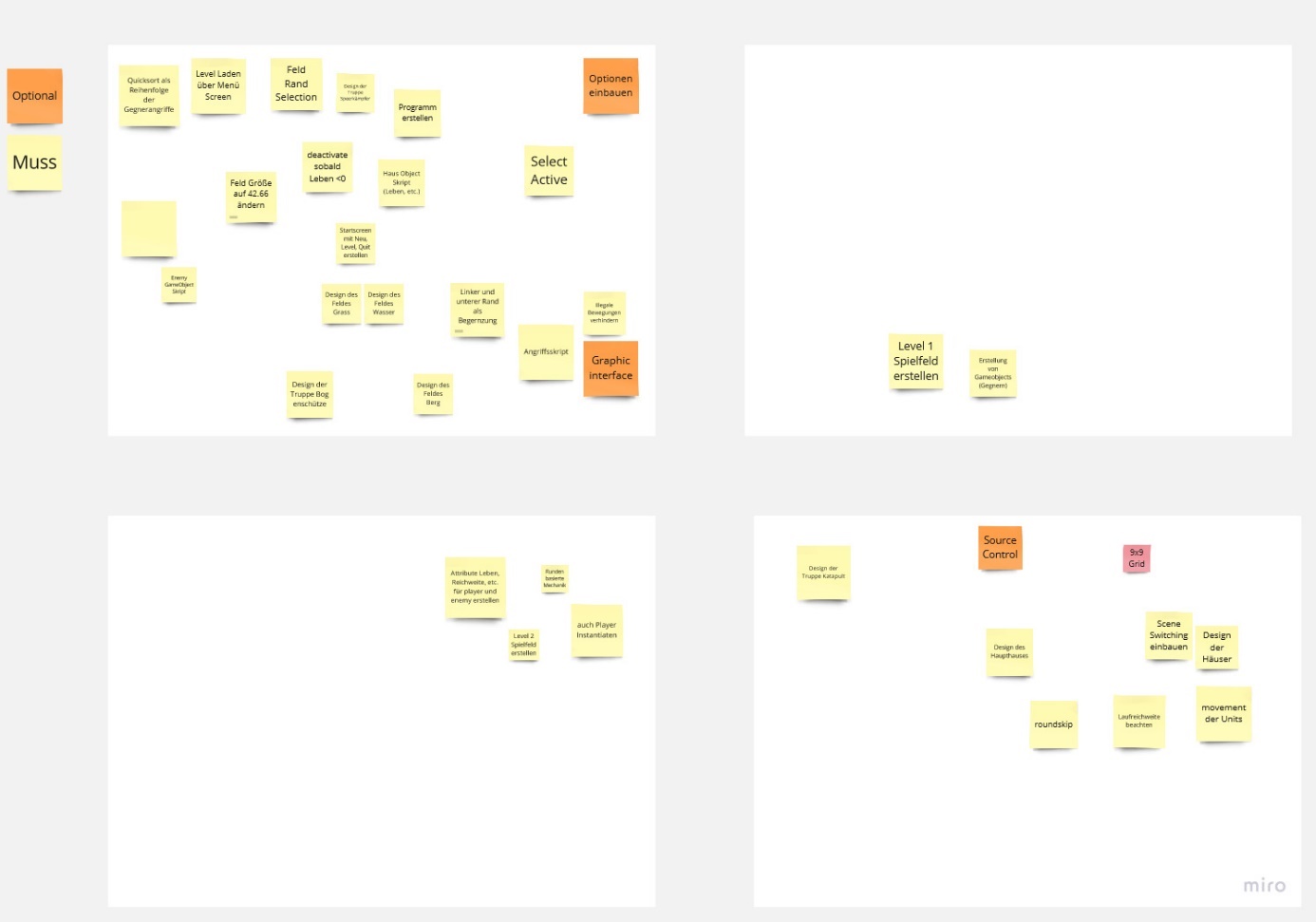
https://answers.unity.com/questions/781870/understanding-unitys-coordinate-system.html 24.03.2023

https://answers.unity.com/questions/799616/unity-46-beta-19-how-to-convert-from-world-space-t.html

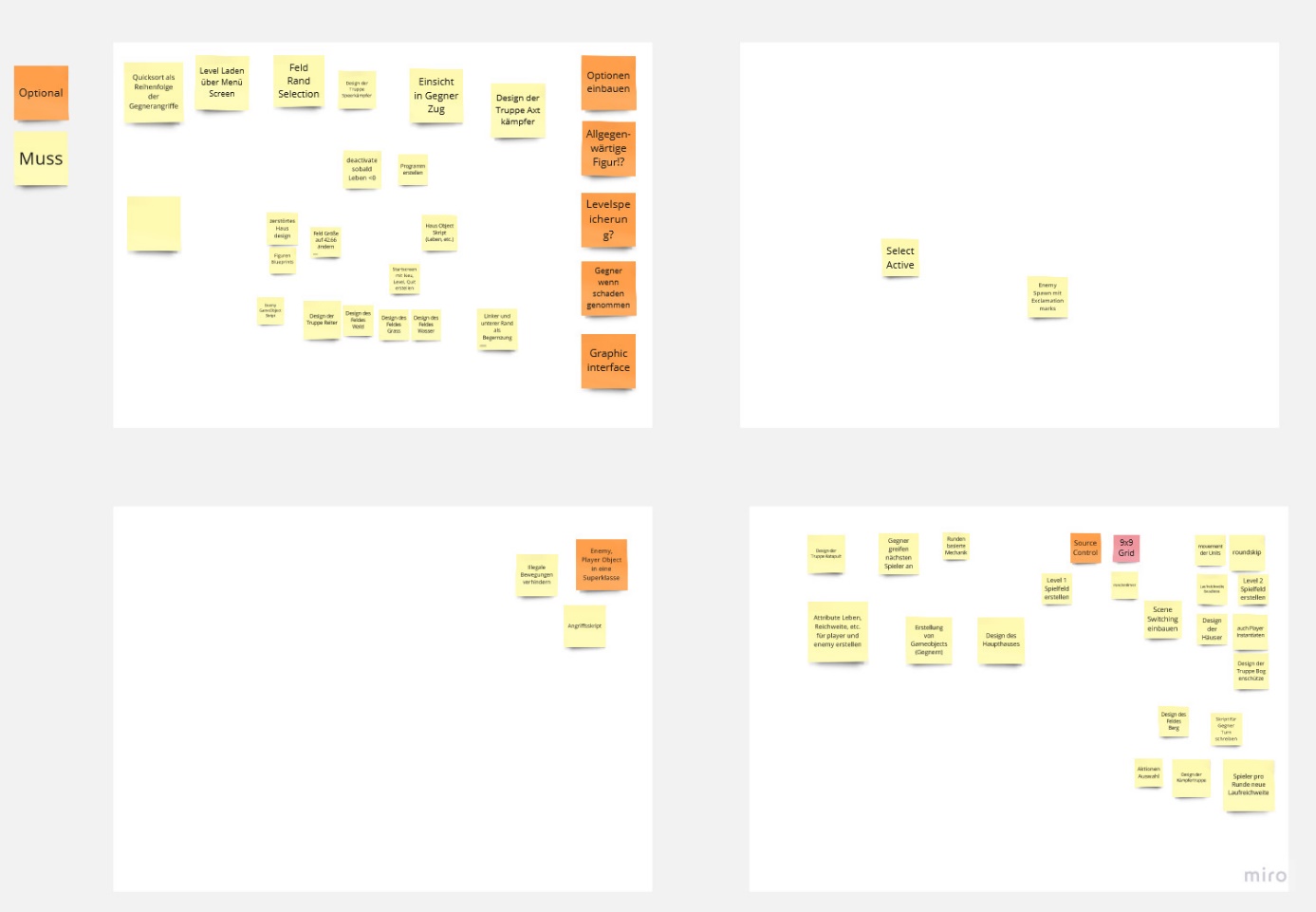
Erstellung des neuen Koordinatensystems, Verwendung der ScreenToWorldPoint und WorldToScreenPoint Funktionen, und die Ermittlung

Dokumentation mit dem Scrum Board:

(Stand 01.03.2023)



(Stand 15.03.2023)



(Stand 24.03.2023)

Ein Bild, das Tisch enthält.

Automatisch generierte Beschreibung

(Stand 31.03.2023)

Ein Bild, das Tisch enthält.

Automatisch generierte Beschreibung

# Literaturverzeichnis

SourceTree Installation und GitHub Einrichtung

https://www.youtube.com/watch?v=EfU4o7U\_xAk, 9.02.2023

https://www.youtube.com/watch?v=adJXt0bN1Ak, 9.02.2023

Tutorial zur Erstellung eines MainMenus

https://www.youtube.com/watch?v=-GWjA6dixV4, 9.02.2023

https://www.youtube.com/watch?v=CGsEJToeXmA, 12.02.2023

Erstellung eines Spielfeldes mit dem Instantiate-Befehl

https://www.youtube.com/watch?v=mqedBCiZN3s, 14.02.2023

https://www.youtube.com/watch?v=u2\_O-jQDD6s, 14.02.2023

Anklicken von Objekten

https://www.youtube.com/watch?v=kkkmX3\_fvfQ, 12.03.2023

Bilddateien

Hintergrund:

https://wallpaperaccess.com/full/38123.jpg, 9.02.2023

Designs der Spieler und Gegner in Anlehnung an:

https://de.cleanpng.com/png-4od85s/, 31.03.2023

https://www.figurenschnapp.de/de/elastolin-7-cm-reiter-mit-signum/adler-nr.-8453-schoene-figur, 31.03.2023

https://www.alamy.de/alter-soldat-mit-speer-zenturio-im-krieg-3d-rendering-image472265077.html?imageid=E586DD14-0F94-4BF4-A600-23B0F179C35A&p=1441946&pn=1&searchId=6649504debe26e66003fb01ca939e9bb&searchtype=0, 31.03.2023

https://www.alamy.de/concept-art-fantasy-abbildung-krieger-jager-mit-speer-image230494139.html?imageid=667034CB-09DA-4B3D-8635-7DB7FE88D955&p=388439&pn=1&searchId=6649504debe26e66003fb01ca939e9bb&searchtype=0, 31.03.2023

https://www.fantasywelt.de/Katapult?curr=EUR, 31.03.2023

https://www.spielfigurenportal.de/Bullyland-80788-Axtkaempfer-rot-9-cm-Welt-der-Ritter, 31.03.2023

https://wiki.kingdoms.com/tiki-index.php?page=Axtkämpfer+%28Germanen%29, 31.03.2023