MMS-Projekt Abschlussbericht

Digitale Spielesammlung mit Fokus auf Multimedia-Aspekte

Victoria Plöckinger Michael Reinegger Stanislaus Silber
Julian Weichinger Thomas Wimmer

1. Beschreibung

Die Spielesammlung verfügt über ein Startmenü mit einfachen Audio- und Grafikeinstellungen. Über eine Spieleauswahl können verschiedene Spiele gestartet werden. Folgende Spiele wurden realisiert: Minesweeper, Breakout, Snake und Pong, Letzteres als Multiplayer-Version. Bei den jeweiligen Spielen können Namen angegeben werden, um einen Eintrag in einer Rangliste zu speichern, um sich somit mit anderen Spielern bzw. mit älteren Spieldurchläufen vergleichen zu können.

Um einen besseren Überblick im Spiel zu erhalten wurden die Spiele, wenn als nötig erachtet mit Timern, Punkteangaben etc. versehen.
Für die Entwicklung wurden herkömmliche Regeln und Spielprinzipien herangezogen und durch einige Ergänzungen erweitert.

2. Entwicklung

2.1. Überblick

Das Projekt wurde mit der Entwicklungsumgebung Unity erstellt und als einzelne 2D-Spiele umgesetzt, die am Ende in einem Menü zusammengeführt wurden. Dazu werden von Unity selbst hilfreiche Tutorials[1] angeboten, in denen schrittweise die Konzepte von Unity erklärt werden.

Programmteile werden dabei in Unity als Prefabs, also als eine Art Interface, realisiert, die schlussendlich im Scene Editor zusammengesetzt werden können. Diese Teile bestehen aus C# (C-Sharp) Scripts. Einige davon zum Beispiel diverse Collision Detections sind bereits vorhanden, andere Scripts müssen selbst geschrieben werden. Außerdem können Resources wie zum Beispiel Sprites und Audios den jeweiligen Prefabs zugeordnet werden, um sie in den Scripts verwenden zu können.

Die Spiele wurden jeweils von einem Teammitglied erstellt. Das erfahrenste Mitglied im Umgang mit Unity kümmerte sich um das Spiele-Hub, die Einstellungen und die jeweiligen Ranglisten.

Mit der Entwicklung wurde am 14. Mai begonnen. Als interne Deadline wurde der 27. Mai gesetzt. Die Entwicklung der jeweiligen Spiele erfolgte parallel. Nach Ende der Deadline wurden keine Features mehr hinzugefügt. Die Spiele wurden zusammengesetzt und das Projekt wurde auf Fehler überprüft, wobei aufgetretene Fehler bestmöglich behoben wurden.

Ein Link zum Projekt, befindet sich bei den Links auf der letzten Seite.

[1] Vgl. “Learn game development w/ Unity | Courses & tutorials in game design, VR, AR, & Real-time 3D | Unity Learn” https://learn.unity.com/ am 01.06.2022

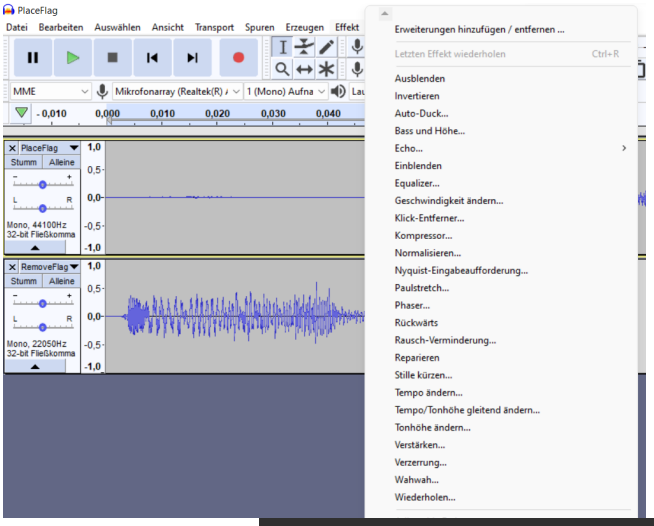
2.2. Entwicklung von Minesweeper

Thomas Wimmer

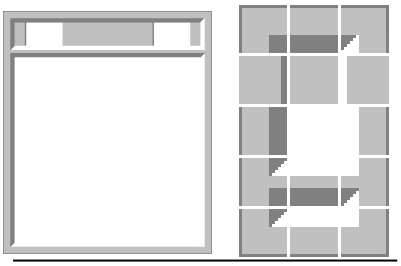
Die Grundkonzepte des Minesweeper Spiels wurden von einem Youtube Tutorial[2] übernommen. Anschließend wurden Fehler behoben und zusätzliche Features eingebaut. Darunter drei Schwierigkeitsstufen - durch einfache Modulo Operation realisiert - eine optionale Flagge, die bei Spielbeginn auf das erste freie Feld gesetzt wird, Timer und Counter, um als Highscore zu dienen, und einiges mehr.

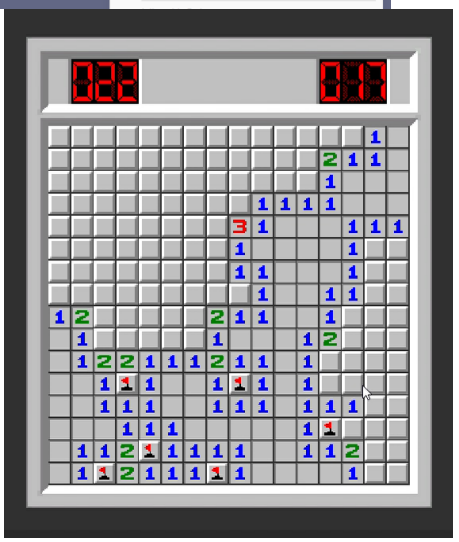
Dann wurden die vorläufigen Sprites durch endgültige Sprites ersetzt. Die Sprites der Minenfelder haben die Größe 16x16 Pixel. Das Spielfeld besteht aus 16x16 dieser Felder somit ist es 256x256 Pixel groß. Um das Spielfeld wurde ein Rahmen mit 16x16 Pixel Sprites erstellt. Diese Rahmen bestand ursprünglich aus nur 8 Sprites, die mittels Tilemap eingefügt werden konnten, um das Spielfeld beliebig vergrößern zu können.

Geräusche wurden mit Audacity[3] aufgenommen und Lautstärke, Geschwindigkeit und Tonhöhe verändert (siehe Abbildung rechts). Diese Geräusche verwendet ein Audio Manager Script[4] um bei bestimmten Events (z.B. dem Anklicken eines Minenfeldes) einen Sound abzuspielen. Insgesamt gibt es sechs verschiedene Sounds, wovon vier selbst erstellt wurden.



Schließlich wurden noch Zähler für verbleibende Flaggen und die abgelaufene Zeit entworfen. Der Hintergrund musste dazu angepasst werden. Dabei wurden aus 8 Sprites 14 Sprites (siehe Abbildung unten). Die Umsetzung mit Tilemap wurde aus zeittechnischen Gründen verworfen. (Abbildung rechts: Fertiges Spiel)





[2] "Let's Make MINESWEEPER in Unity" https://www.youtube.com/watch?v=83vBzeeRM-c am 01.06.2022

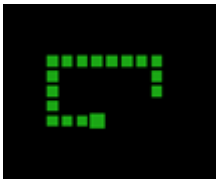
[3] https://www.audacity.de

[4] "Introduction to AUDIO in Unity" https://www.youtube.com/watch?v=6OT43pvUyfY am 03.06.2022

2.3. Entwicklung von Snake

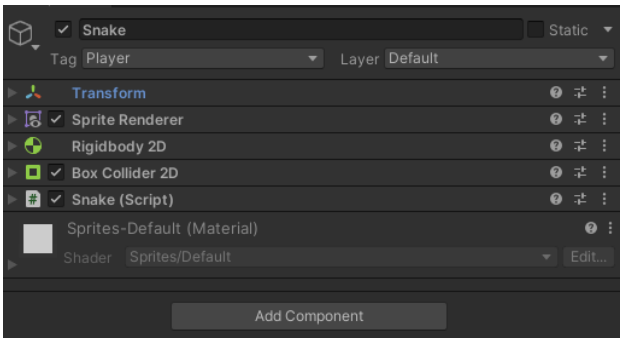
Victoria Plöckinger

Snake wurde basierend auf einem Tutorial[5] realisiert, um die Grundidee des Spiels zu verwirklichen. Weitere Ideen für den Spielinhalt wurden später eigenständig implementiert.



Zuerst wurde ein 2D-Sprite für den Kopf der Schlage, den Snake-Kopf, hinzugefügt und adaptiert. Dieses Objekt wurde durch eine Rigidbody Komponente zu einem beweglichen Element in der Szene.
Ein Box-Collider überprüft auf mögliche Zusammenstöße zwischen Kopf und anderen Elementen, wie etwa Wänden oder Äpfeln. Auf diese Events reagiert die Snake entsprechend. Der Körper der Snake besteht aus kleineren Kopf Elementen, die hinten angefügt werden, wenn ein Apfel oder ein Double-Up gegessen wird.

Die Wände und der Apfel sind in gleicher Weise wie der Kopf erstellt worden, wobei beide keinen Rigidbody benötigen und die Wände durch Transform und Skalierung noch in die richtige Größe gebracht wurden. Anschließend wurde für jedes Objekt ein C#-Script hinzugefügt (siehe Abbildung rechts: Snake Objekt mit Komponenten).



Die Bewegung des Snake-Körpers wurde mit Hilfe von den Methoden Update() und FixedUpdate() erzeugt. Der Input wird einmal pro Frame abgefragt (Update), währenddessen die Reaktion der Snake beliebig oft in den Frames abgefragt wird (FixedUpdate), um so die Bewegungsabläufe auszuführen.

Äpfel, Gift und Double-Up - letztere beide wurden zusätzlich zu der herkömmlichen Variante eingebaut - , im Folgenden Consumables genannt, werden innerhalb eines Rahmen Objektes zufällig platziert. Dieses Rahmen Objekt wird in einem Food-Script referenziert, um so die Position der Consumable Elemente zu berechnen.

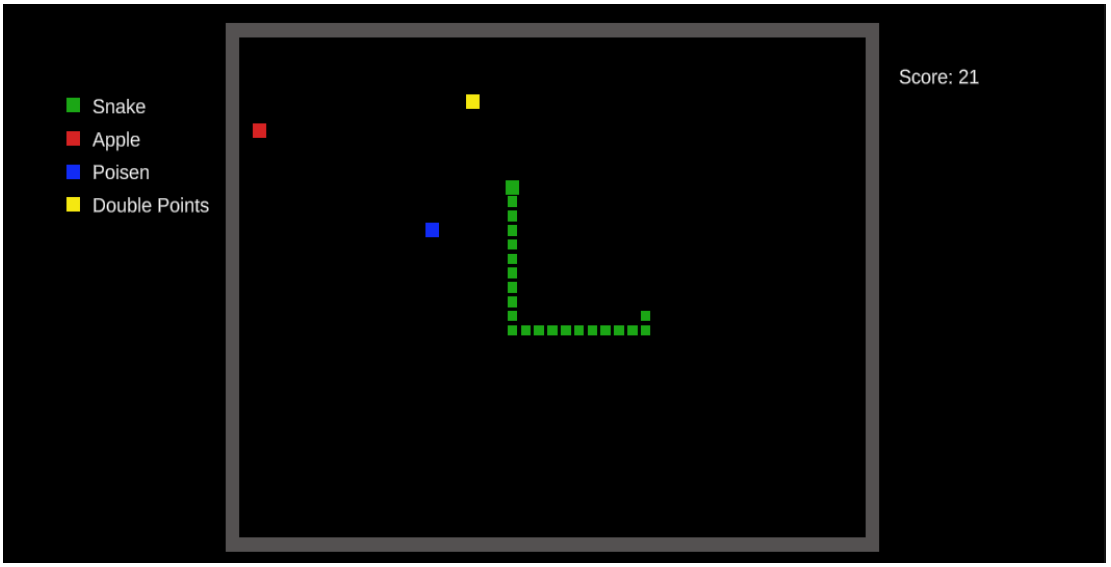
Die Box Collider Komponente der Consumables ermöglicht es festzulegen, was passiert, wenn der Snake-Kopf mit diesen zusammenstößt. Um zwischen Äpfeln, Gift und Double-Ups zu unterscheiden werden entsprechende Tags benutzt. Im Skript führt dann der jeweilige Tag zu einer passenden Reaktion: Wachsen, Game Over oder zwei zusätzliche Punkte. Dabei wird die Position der Consumable Elemente auf Null gesetzt, so dass sie scheinbar verschwinden.

[5] "How to make Snake in Unity" https://www.youtube.com/watch?v=U8gUnpeaMbQ&t=724s am 02.06.2022

Berührt die Snake eine Wand oder isst sie Gift, wird das Spiel auf die Ausgangssituation zurückgesetzt und der Score beginnt wieder bei null. Das Spiel ist vorbei.

Eine minimal gehaltene Legende wurde als Hilfeleistung für den Spieler als auch zur Verbesserung der Übersichtlichkeit als notwendig erachtet und implementiert.
Der Score war eine der größten Herausforderungen, da hierfür ein UI-TextMashPro-Objekt notwendig war, welches dann in C# ausprogrammiert werden musste. Hierzu gibt es jedoch momentan nur Tutorials mit einem UI-Text Objekt, da dies erst kürzlich von Unity verändert wurde. Für das UI-TextMashPro-Objekt muss im Code TMPro importiert werden und statt einem Text-Objekt ein TextMeshProUGU-Objekt erstellt werden. Dies herauszufinden hat einiges an Zeit beansprucht.

Abschließend wurde noch ein angenehmer Song als Hintergrundmusik gewählt.
(Abbildung unten: Fertige Implementierung)



2.4. Entwicklung von Pong

Julian Weichinger

Die Umsetzung der Grundidee wurde mithilfe eines einfachen Youtube Tutorials[6] realisiert. Das Augenmerk lag bei diesem Projekt auf dem Erscheinungsbild und der Interaktivität des Spiels.

Die benötigte Punktezahl damit ein Spieler gewonnen hat, kann am Beginn zwischen 5, 10 und 15 Punkten ausgewählt werden. In diesem Textfeld befindet sich ebenfalls eine Beschreibung der Steuerung. Eine besondere Schwierigkeit bestand darin, die Vektoren zu bestimmen mit denen der Ball am Anfang von der Mitte aus weggeschossen wird. Da dieser am Anfang zufällig bestimmt wurde, konnte es passieren, dass der Ball zu schnell oder viel zu langsam war. Dieses Problem konnte gelöst werden, indem diskrete Werte in ein Array verpackt wurden und auf dieses mittels einem zufälligen Index zugegriffen wird.
Es kam auch des öfteren vor, dass der Ball bei zu hoher Geschwindigkeit aus dem Spielfeld verschwand, was sich durch Erhöhung der Reibung am Ball und der Begrenzung weitestgehend eindämmen ließ.

Die Punktzahl wird durch TextMeshPro Felder dargestellt und auch entsprechend aktualisiert. Erzielt ein Spieler die voreingestellte Punktzahl erscheint ein Textfeld, welches dem Spieler zu seinem Erfolg gratuliert.

Nachdem das Grundgerüst stand, wurde sich dem Design des Spiels gewidmet, welches mit dem Open Source Bild-bearbeitungs-programm GIMP[7] verwirklicht wurde. Ziel war es dem Atari-Klassiker ein helles und modernes Auftreten zu verpassen.

[6] https://www.youtube.com/watch?v=PteH1y9SC84&t=545s am 1.6.22
[7] https://www.gimp.org/

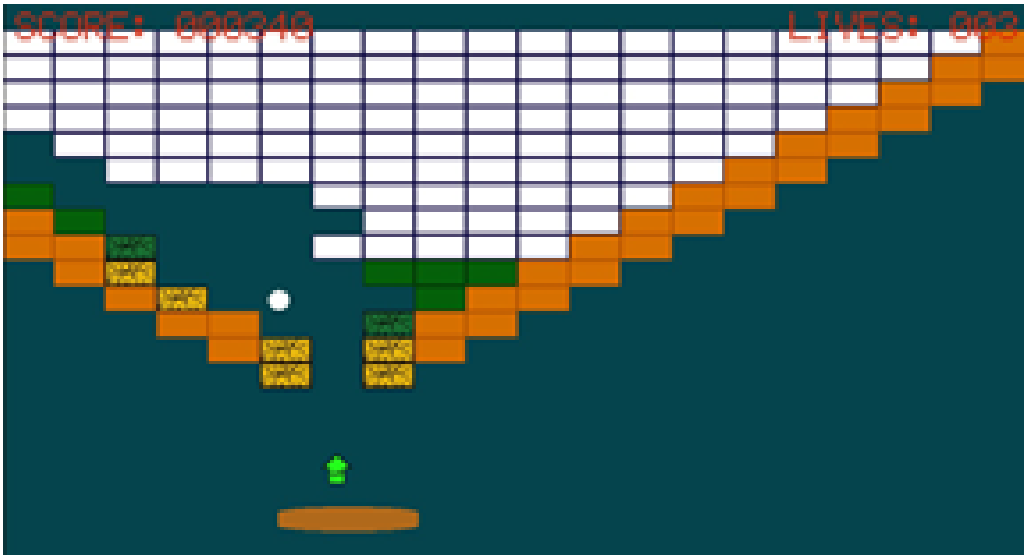
2.5. Entwicklung von Breakout

Stanislaus Silber

Breakout beschreibt in dieser Umsetzung wie sonst auch üblich ein Spiel, in dem man mit einem frei beweglichen Balken einen Ball davon abzuhalten versucht, am unteren Bildschirmrand zu entkommen, während dieser an besagtem Balken, den übrigen drei Wänden als auch an im Spielfeld verteilten Blöcken abprallt. Weiters werden eben jene Blöcke bei Treffern durch den Ball zerstört. Das Spiel ist gewonnen wenn alle Blöcke zerstört wurden.

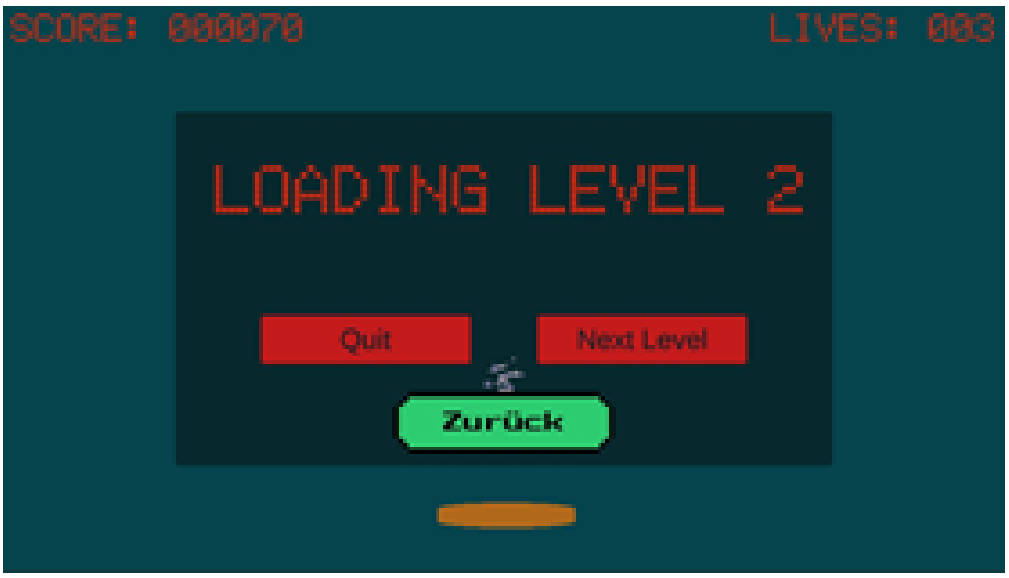
Neben diesen Basis-Features enthält unsere Version:

- Blöcke in unterschiedlichen Farben mit unterschiedlichen Anzahl der Treffer um sie zu zerstören als auch dementsprechend höhere Punkte als Belohnung. Bereits getroffene aber nicht zerstörte Blöcke werden außerdem durch Risse hervorgehoben.
- Optischer Effekt bei der Zerstörung eines Blocks (Pixel-Explosion)



- „Lebens-" Power-Ups, welche selten und zufällig bei der Zerstörung eines Blockes erzeugt werden und vom Spieler mit dem Balken gefangen werden können um ein zusätzliches Leben zu bekommen.
- Ein Punktesystem mit Namenserfassung zum Vergleich mit anderen Spielern.
- 5 spielbare Levels und man schreitet in höhere bzw. schwierigere Levels vor wenn alle Blöcke zerstört wurden.
- Eine akustische Untermalung mit einem Background-Song und zwei unterschiedlichen Soundeffekten bei je einem Block-Treffer als auch dessen Zerstörung.

- Eigenes Overlay nach jedem Level bzw. bei „Game Over" mit Optionen zum Beenden und Laden des nächsten Levels bzw. Neubeginn des Spiels.



Rein aus Demonstrationszwecken ist einerseits den 5 eigentlichen Levels noch ein Level mit 3 unterschiedlichen Blöcken vorgereiht und andererseits die Levelabfolge zufällig eingestellt. Dies würde sich für eine eventuelle Distributionsversion noch ändern.

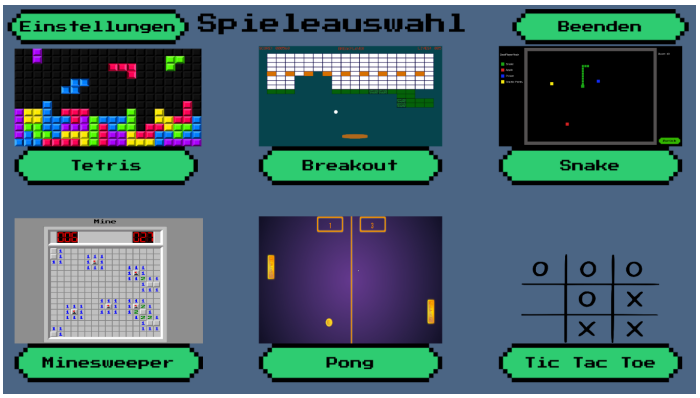
Wir konnten uns für die Umsetzung vieler der einzelnen Objekte angenehmerweise auf die Physics-Engine verlassen, und dabei mit dem Event-Listener und Collidern auf bestimmte Situationen reagieren. Auch die Einbindung der akustischen Elemente verlief dank Unity reibungslos. Ein weiterer Grund für die Wahl von Unity im generellen war natürlich auch die unglaublicher Erleichterung die durch das Prefab-System mit einhergeht, auch wenn diese dann zu einer Komplikation führten. Alle notwendigen Sprites wurden ebenfalls in GIMP[8] realisiert, was dank vorhandener Vorkenntnisse ein Leichtes war.

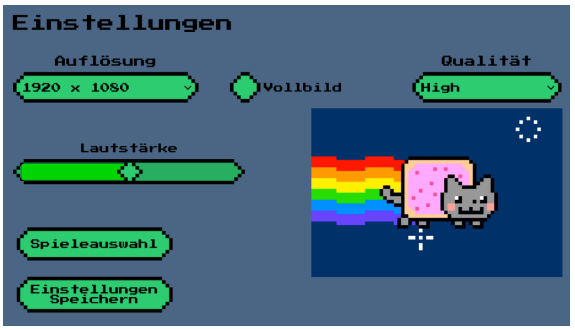
[8] https://www.gimp.org/

2.6. Entwicklung des Hauptmenüs

Michael Reinegger

Das Hauptmenü[9] bietet die Möglichkeit, zwischen sechs verschiedenen Spielen zu wählen. Davon wurden wie bereits einleitend erwähnt nur vier umgesetzt und die anderen dienen lediglich als Platzhalter. Des Weiteren kann man auf das Einstellungsmenü zugreifen.



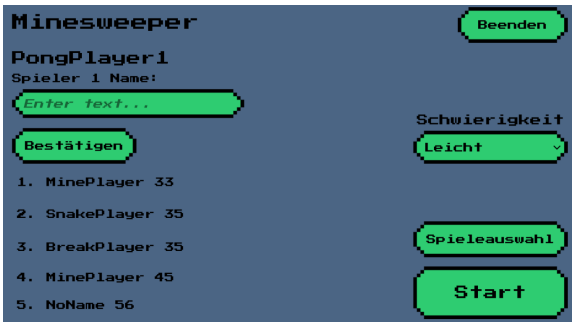


Im Einstellungsmenü können sowohl die Auflösung als auch simple Grafik-Einstellungen sowie das Aktivieren und Deaktivieren des Vollbildmodus eingestellt werden. Ebenso kann die Lautstärke der Hintergrundmusik und der Soundeffekte angepasst werden.
Diese Einstellungen können gespeichert werden, und sind somit automatisch beim nächsten Start der Spielesammlung aktiv.

Wenn man sich nun für ein Spiel entschieden hat, erscheint ein Menü in welchem man Einstellungen, die für das jeweilige Spiel relevant sind, vornehmen kann und das Leaderboard[10] wird angezeigt.
Bei Breakout und Snake kann jeweils der Name des Spielers ausgewählt werden und die besten fünf Spieler werden mit Highscore angezeigt.

[9] "How to Create a Settings Menu in Unity - Simple Talk" https://www.red-gate.com/simple-talk/development/dotnet-development/how-to-create-a-settings-menu-in-unity/ am 03.06.2022
[10] "How to keep score in Unity (with loading and saving) - Game Dev Beginner" https://gamedevbeginner.com/how-to-keep-score-in-unity-with-loading-and-saving/ am 03.06.2022

Minesweeper hingegen bietet auch die Möglichkeit, die Schwierigkeit zwischen Leicht, Normal und Schwer zu wechseln. Es werden jeweils die fünf Spieler angezeigt, die das Minenfeld am schnellsten aufdecken konnten, mit ihrer Zeit in Sekunden. Die Anzeige der Highscores ist abhängig von der ausgewählten Schwierigkeit.



Das Multiplayer-Spiel Pong bekam ebenfalls ein anderes UI. Hier können die Namen für den linken und den rechten Spieler ausgewählt werden. Das Scoreboard zeigt hier jedoch nur die letzten fünf Spiele an.

Wenn man nun mit den Einstellungen zufrieden ist, kann man mit dem Startknopf das jeweilige Spiel starten. Die Spiele wurden jeweils von einem Teammitglied erstellt und mussten somit in das Hauptmenü Projekt eingebunden werden.

In Unity ist es zwar relativ einfach ein anderes Projekt zu importieren, da man einfach alle Assets(Sprites, Sounds, Skripte, etc.) in den Projektordner einfügen und dann die jeweilige Scene öffnen kann. Da jedoch die Spiele alle leicht anders entwickelt wurden und eine andere Ordnerstruktur als das Hauptmenü verwendeten, mussten alle Sprites, Sounds und Skripte den Gameobjekten neu zugewiesen werden.
Außerdem kam es beim Öffnen der Szenen der Spiele in dem neuen Projekt oftmals zu Verschiebungen der Gameobjects, die alle wieder korrigiert werden mussten.

Bei jedem Spiel wurde auch ein UI-Element eingefügt, um den Spielernamen anzuzeigen und ein Zurück-Knopf, um wieder zum Menü zu gelangen. Bei Breakout und Pong wurde noch jeweils ein Pause Menü eingebaut.

Etwas Code wurde je Spiel hinzugefügt, der entwender nach Beenden eines Spiels oder beim “Game Over” ausgeführt wird, der die alten Scores ausliest und den neuen Score, falls notwendig, einfügt und speichert. Diese Code-Segmente konnten mit lediglich minimalen Änderungen für alle Spiele adaptiert werden. Das Speichern und Auslesen der Scores wird mithilfe der PlayPrefs von Unity übernommen. Diese Lösung ist nicht ideal, da die Scores unter Windows in den Registry gespeichert werden, aber ausreichend für ein Projekt dieser Größe.

3. Zusammenfassung

Als größte Schwierigkeit bei der Realisierung des Projekts kann hier bedenkenlos die Unvertrautheit mit der gewählten Engine, also Unity genannt werden. So leicht und angenehm unser kurzer Exkurs von Java nach C# war, umso herausfordernder war der Neuanfang in Unity. Hier war vor allem die breite Auswahl an Tutorial Videos zu Unity selbst hilfreich.

Leider wirkte sich unser Erfahrungsdefizit mit Unity nicht nur im reinen Programmieraufwand sondern auch in der Fehleranfälligkeit bzw. im Zeitaufwand für die Fehlersuche aus. So schlich sich beispielsweise bei Breakout ein Fehler ein, als bei einem der Blöcke der Collider zugeklappt wurde, sodass dieser als gelöscht angenommen wurde. Nach der vermeintlichen Korrektur waren plötzlich zwei Collider vorhanden. Dies in Kombination mit dem Prefab-System resultierte im weiteren Verlauf also darin, dass sich manche Blöcke wider ihrer Trefferanzahl verhielten und doppelt so schnell zerbrochen werden konnten als vorgesehen.

Einige Fehler hatten auch damit zu tun, dass aufgrund von zu hoher Geschwindigkeit der Gegenstände, diese sich auf einmal außerhalb des Spielfelds befanden und nicht mehr gesehen werden konnten. Das Auffinden der richtigen Skalierung war dabei keine einfache Aufgabe, die Spiele sollten flüssig und nicht langweilig, aber keinesfalls unspielbar sein. Viele Bugs wurden erst durch die Revision des Projektes *von* beziehungsweise *mit* unserem Unity-vertrautesten Kollegen gefunden und entschärft. Jedoch traten besonders beim Einbinden der Spiele in das Hauptmenü unerwartete Fehler auf; Sprites mussten neu bearbeitet werden, Sound konnte nicht importiert werden u.Ä.

Bestimmte Eigenschaften der Spiele konnten aufgrund von der gesetzten Deadline nicht mehr realisiert werden, wobei in den meisten Fällen die Spiele noch um diverse Soundeffekte und zusätzliche Optionen erweitert hätten werden sollen.

3.1. Rückblick

Viele der angeführten Probleme wären für erfahrener Unity-Anwender zeitlich wohl weniger schwer ins Gewicht gefallen als dies bei uns der Fall war. Wir hätten dann diesen Zeitgewinn in eine deutliche Erweiterung des Spieles, wie in etwa weitere PowerUps, zufällig generierte Levels oder möglicherweise sogar einen Multiplayer Modus, steckten können. Wir haben also, wenn schon nicht erlernt, dann zumindest an der eigenen Haut erfahren wie stark sich domänenspezifisches Wissen auf ein Projekt auswirken kann.

Dennoch wagten wir uns an Sound, Tilemaps, Interaktivität und Einstellungen heran, um so die Spiele wenigstens nach außen hin attraktiv zu machen. Somit war es schließlich ein doch sehr lehrreiches Projekt.

4. Weiterführende Links

Projekt im Github-Repository

- https://github.com/MrCodeEU/MMS-Projekt-Spielesammlung

Erste Spiele in Unity programmieren:

- https://learn2programe.github.io/learn2proGrAME-Tutorial/
- https://learn.unity.com/
- https://www.youtube.com/c/unity/videos
- https://www.youtube.com/playlist?list=PLPV2KyIb3jR5QFsefuO2RlAgWEz6EvVi6

Open Source Tools für Coding, Audio- und Bildbearbeitung:

- Unity: https://unity.com/de
- Visual Studio: https://visualstudio.microsoft.com/de/
- Audacity: https://www.audacity.de/
- GIMP: https://www.gimp.org/
- Paint.NET: https://www.chip.de/downloads/Paint.NET_13015268.html