Projektdokumentation

Modul: Projekt Systementwicklung beziehungsweise Projekt Multimedia WS 20/21

Implementierung eines Multimedia Billard Tisches

Gliederung

1. Projektstart

Ausgangssituation

Bevor wir mit der Einteilung in feste Gruppen und Rollen und der Erarbeitung eines Hauptziels beginnen konnten mussten wir uns erst einmal in Unity, C-Sharp und in das vorhandene Projekt einarbeiten beziehungsweise uns einen Überblick verschaffen.
Als dies geschafft war, haben wir uns in der zweiten Woche des Projektes mit der Bestimmung dieser Rollen/Gruppen und Ziele beschäftigt.

Diese Themen werden in den nächsten Absätzen aufgezeigt und näher erläutert.

Hauptziel

Wir als Projektgruppe haben es uns zum Hauptziel gemacht, die Anwendung performanter, stabiler und von der Architektur her besser abtrennbar zu gestalten.

Der Hintergrund des Hauptziels ist, dass die Erstellung dieser Anwendung auch noch von Nutzen sein soll, selbst wenn das Projekt wieder vor Ort an einem realen Multimedia Billard Tisch durchgeführt werden kann.

Gruppen- und Rollenverteilung

Position	Person
Stakeholder	Björn Frömmer
Product Owner	Carolin Hochmuth
Head-of-Tech	Julian Manß
Scrum Master (Graphics)	Laure Hontabat-Franky
Team Graphics	Florian Grimm, Laure Hontabat-Franky, Sebastian Schachner
Scrum Master (Simulator)	Özlem Aydin
Team Simulator	Megi Grabocka, Özlem Aydin, Ivan Dukov, Seda Erdogan, Carolin Hochmuth

Scrum Master (Tracking)	Christoph Meyer
Team Tracking	Christoph Meyer, Julian Manß
Scrum Master (Game)	Samia Bouguar
Team Game	Abdurrahman Yigen, Lorik Gashi, Leon Schrabeck, David Bahcecioglu, Samia Bouguar, Domenico Spadavecchia
Gitlab-Maintainer	Julian Manß und Christoph Meyer

Wir haben uns bei dem verteilen der Rollen und Gruppen die Worte und Anregungen der vorherigen Gruppe zu Herzen genommen und haben von Anfang an einen Product Owner bestimmt, der auch über das gesamte Projekt hinweg gleichgeblieben ist.
Dieser war mit folgenden Aufgaben betreut: Organisation, Dokumentation (Sprint Reviews/ Retrospektiven etc. auf dem Repository), pflegen des Product Backlog (Ticketsystem in Gitlab), Kommunikation zwischen den Teams, Vermittlung von Hilfe bei jeglichen Themen, Aufklärung von Missverständnissen und auftretenden Problemen, dauerhafte Überprüfung und Anpassung des Projektstands.

Des Weiteren haben wir uns für die Notwendigkeit einer Head-of-Tech Rolle entschieden, da unsere Projektarchitektur komplexe Themen beinhaltet und es eine Person geben sollte die hier fachlich zur Seite stehen kann.

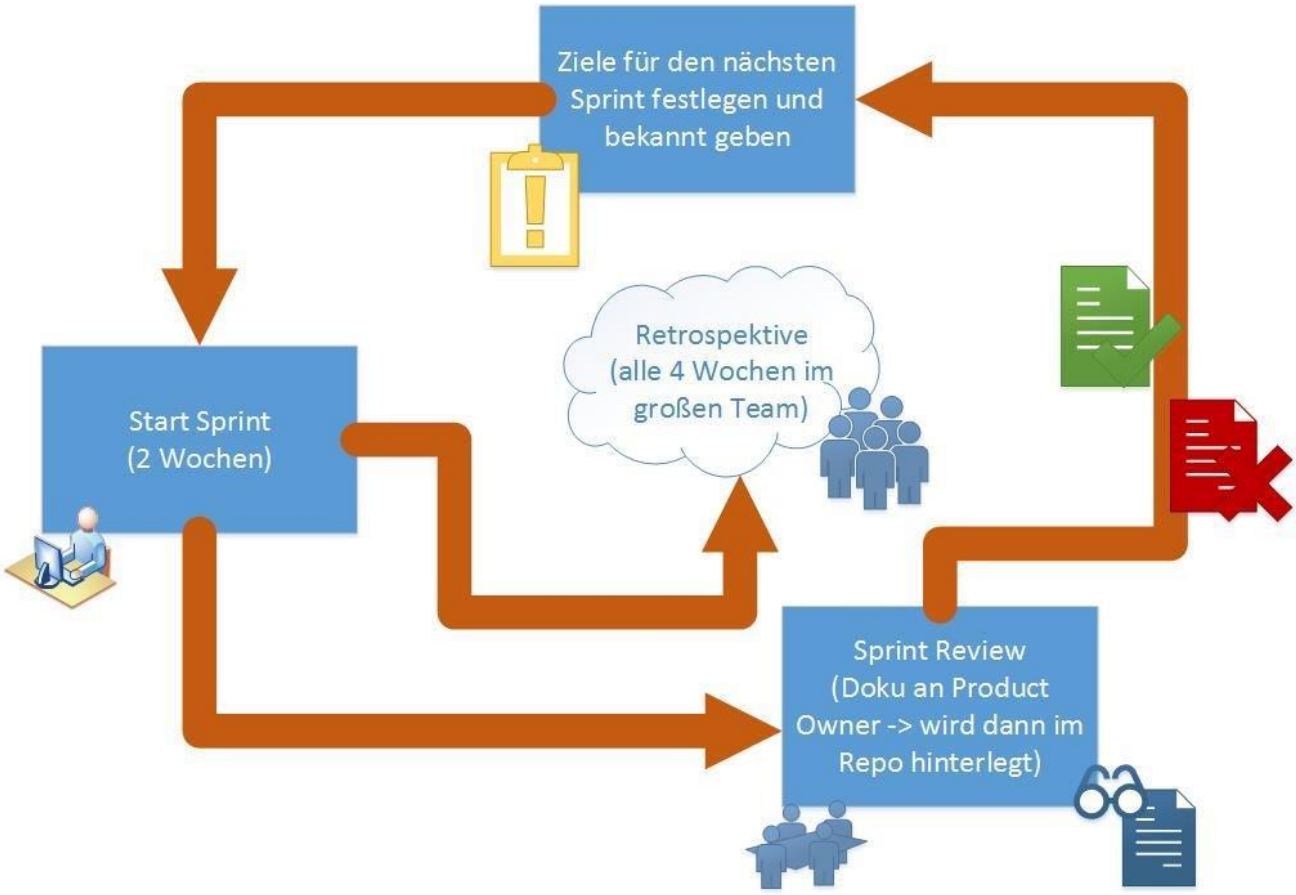
Die Rolle der Gitlab-Maintainer haben wir vergeben, da in diesem Projekt Studenten vom dritten bis zum fünften Semester mitarbeiten und noch nicht jeder sehr gut mit Git arbeiten kann. Jedoch kann eine unvorsichtige oder unbeholfene Arbeitsweise mit Git schnell dazu führen, dass das ganze Projekt stillsteht und repariert werden muss. Deshalb empfehlen wir diese Rolle besonders.
Weitere Informationen zu unserem Aufbau des Git-Repository ist weiter unten zu finden.

Kommunikationsmittel

- Discord-Server PSE Multimedia Billard Tisch mit Channels und Tags der Personen zur Organisation innerhalb der Teams und des gesamten Projekts
- Gitlab zur Dokumentation der Organisation, des Product Backlog und der Meilensteine
- Big-Blue-Button der Hochschule

Workflow

- Sprint Dauer: zwei Wochen
- Sprint Review: immer mittwochs am Ende eines Sprints
- Retrospektive: alle zwei Sprints am Mittwoch, organisiert von den Scrum Mastern
- Illustration:

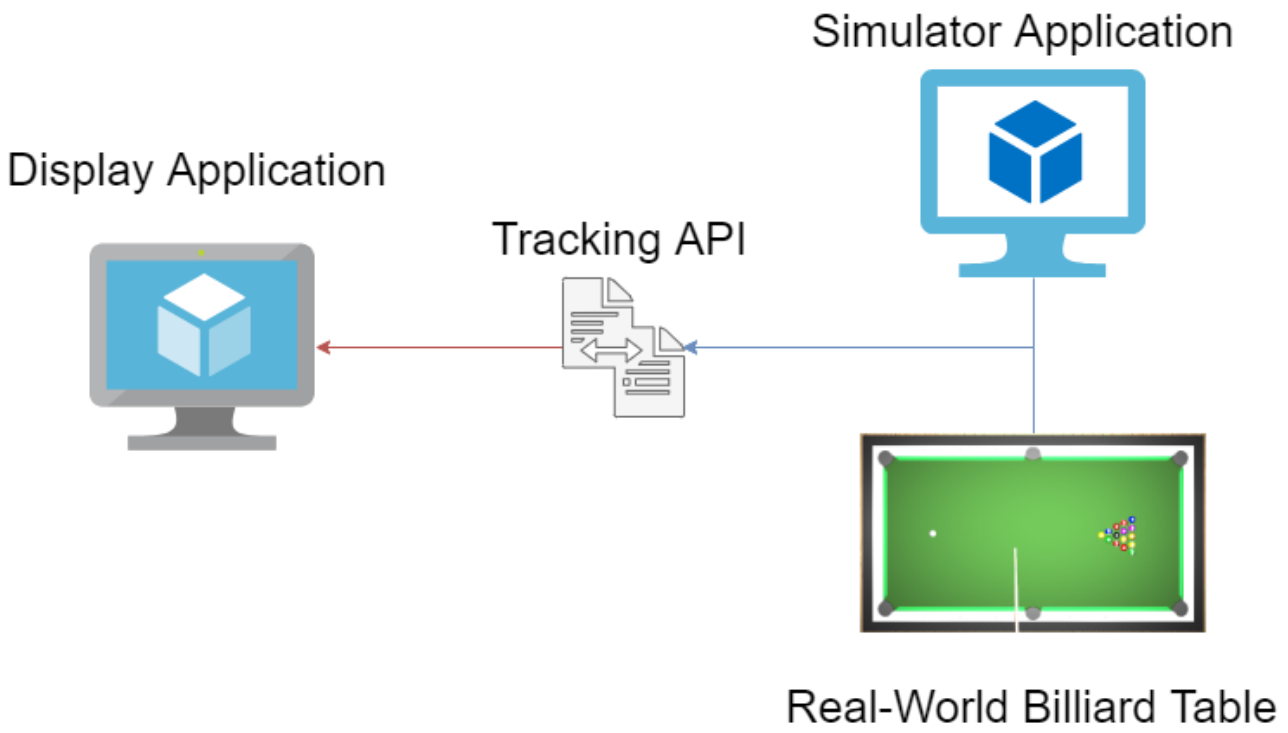


2. Kommunikation der Teams miteinander

Zur Abstimmung der Teams untereinander wurden meist die Channel und Besprechungsräume auf dem Discord Server genutzt. Des Weiteren fanden viele Absprachen bei den wöchentlichen Veranstaltungsterminen statt. Durch diese hohe Anzahl an Kommunikationsschnittstellen konnten viele Missverständnisse im Projekt vermieden werden. Sollte es doch mal zu Missverständnissen zwischen den Teams gekommen sein wurde dies noch rechtzeitig erkannt und zwischen den Teams, mit Hilfe des Product Owner und des Head-of-Tech, geklärt.

3. Projektarchitektur

Abbildung der Architektur:



Wie in der Abbildung zu sehen ist haben wir uns bewusst dafür entschieden zwei eigenständige und voneinander unabhängige Anwendungen zu erstellen. Zum einen den Simulator und zum anderen das Display. Der Simulator ist die Anwendung, auf der man das Spiel durchführt. Das Display ist die Anwendung, welche die Vorgänge auf dem Simulator eins zu eins spiegelt und die Spielregeln für 8-Ball und 9-Ball auswertet und ausgibt.
Das spiegeln der Simulation auf die Display Anwendung, erfolgt durch die Echtzeit-Aufnahme der Daten über die Tracking API. Der Vorteil dieser Architektur ist, dass die Display Anwendung mit einer beliebigen Quelle der Echtzeitdaten arbeiten kann. Statt eines Simulators kann ein realer Billard Tisch problemlos als Quelle fungieren.

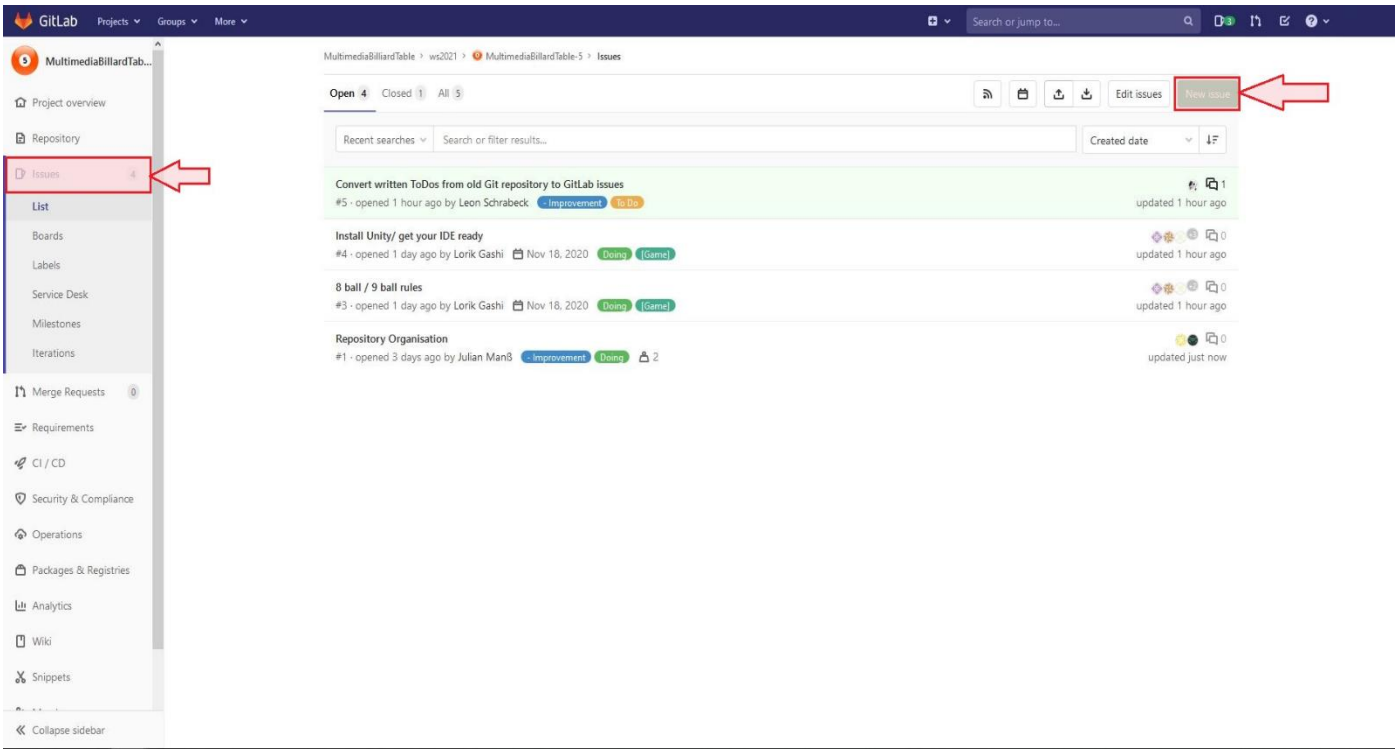
Genaueres dazu findet man in der Dokumentation des Team Tracking.

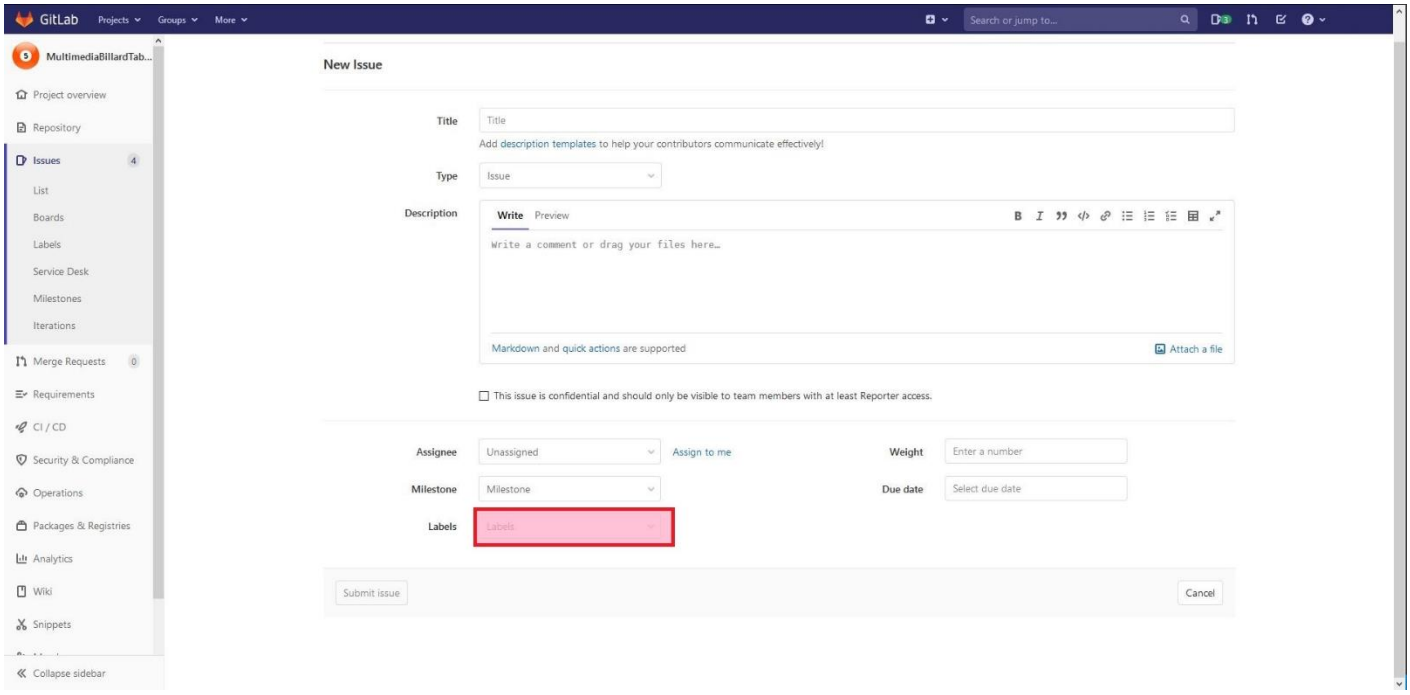
4. Aufbau des Git-Repository

Richtlinien

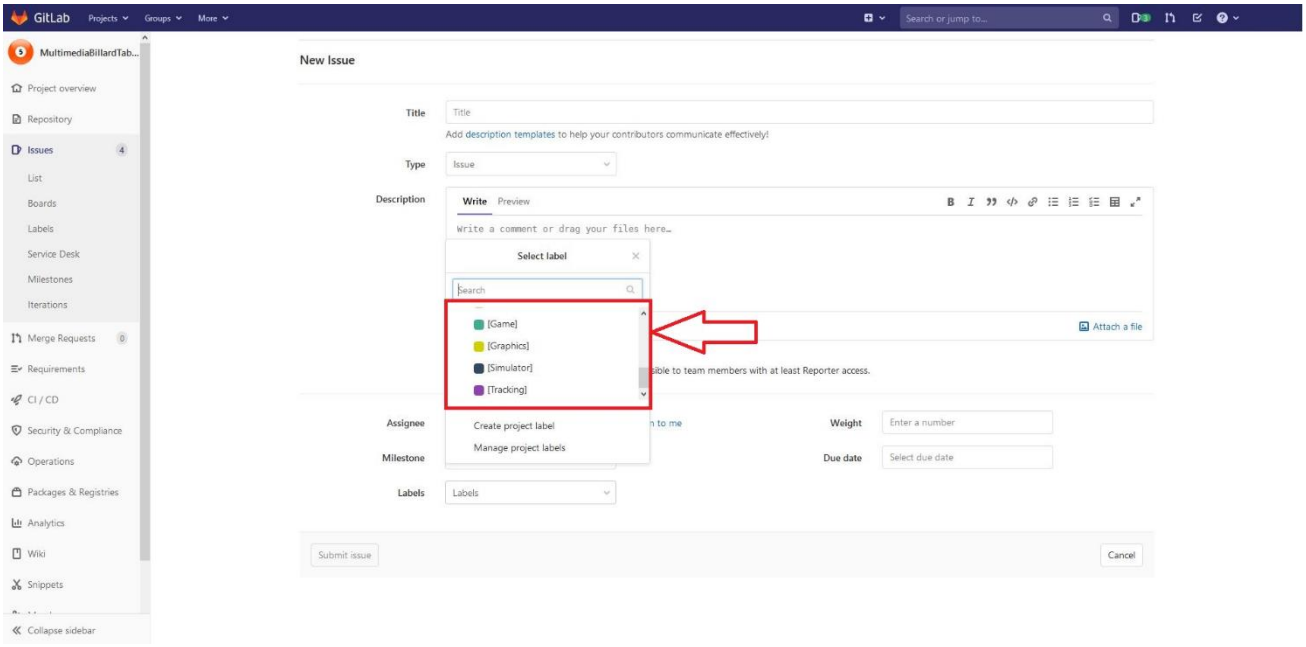
- Gitlab Richtlinien:

  ➔ Wie erstellt man ein Issue:





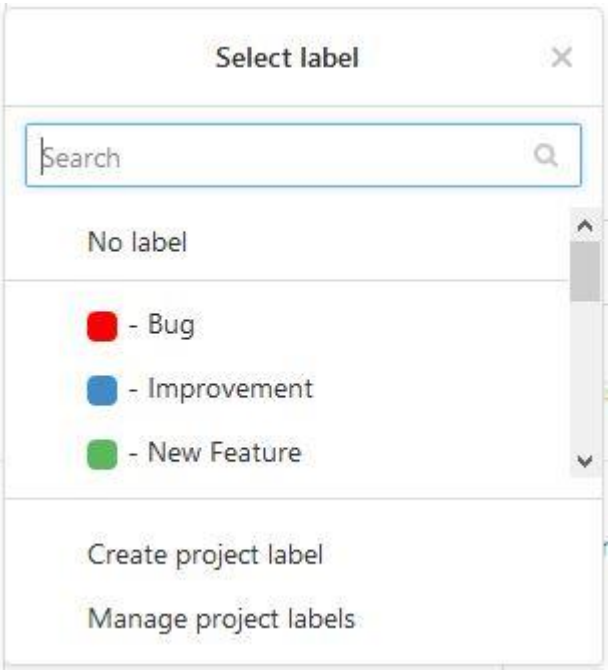
Merke: Es muss immer ein Team-Label hinzugefügt werden. Teamlabels sind in [] Klammern eingeschlossen. Beispiel: [MyTeamName]



Hinweis:

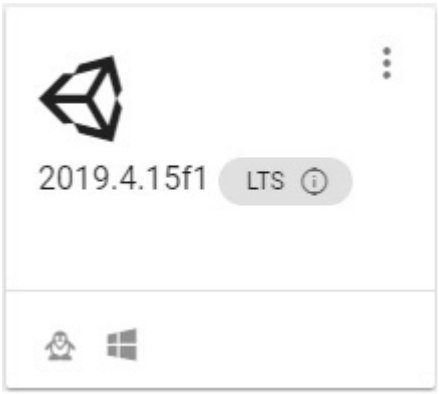
Um den Typ des Problems zu bestimmen sollte die Beschreibungen Aussagekräftig sein. Es ist hilfreich, den Überblick über alles zu behalten, was getan werden muss.

Beschreibende Beschriftungen beginnen mit einem „ – „ zum Beispiel: - Bug



- Git (How-To-Commit):
  - Commits sind immer zugehörig zu einem Ticket.
  - Die Ticketnummer findet man im Ticket welches im Product Backlog erstellt wurde.
  - Commits zeigen an was man gearbeitet hat. Wenn man an mehreren Tasks arbeitet sollte man dafür auch mehrere Commits schreiben.
  - Commits werden auf Englisch geschrieben.
  - Wie ein Commit aussehen sollte: #Ticketnummer, Kurze Beschreibung an was man gearbeitet hat
  - Commit Nachrichten die nicht dieser Regel Folgen werden abgelehnt
  - Commit-Nachricht Beispiel:

- Unity:
  - Im Projekt haben wir die Unity Version 2019.4.15f1 genutzt



  - Unity kann manchmal Merge Konflikt hervorrufen die schwer zu lösen sind. Diese Art von Konflikten entstehen zwischen Unity Scene-Files (Files with *.unity / *.unity.meta Extension). Normalerweise sollte man versuchen solche Merge Konflikte durch eine gute Kommunikation im Team zu vermeiden. Falls jedoch trotzdem passiert kann man diese Probleme mit Hilfe „Smart Merge Tool“ von Unity lösen.

Anwendungen

Die ausführbaren Dateien sind unter dem Ordner „Releases“ unseres Projektes abgelegt.

Sprint Reviews und Retrospektiven

Alle Sprint Reviews und Retrospektiven sind zu finden auf dem Git-Repository unter dem Punkt „Sprint Review/Retrospective Documentation“.

Code Dokumentation

Die Code Dokumentation ist als HTML unter „docs/documentation“ zu finden. Im Git-Repository ist der Link hinterlegt. Git kann diese Offline-Dokumentation nicht darstellen, deshalb muss man die index.html lokal auf dem PC öffnen.

5. Projektdokumentation aller Teams

Projekt Dokumentation: Team Graphics

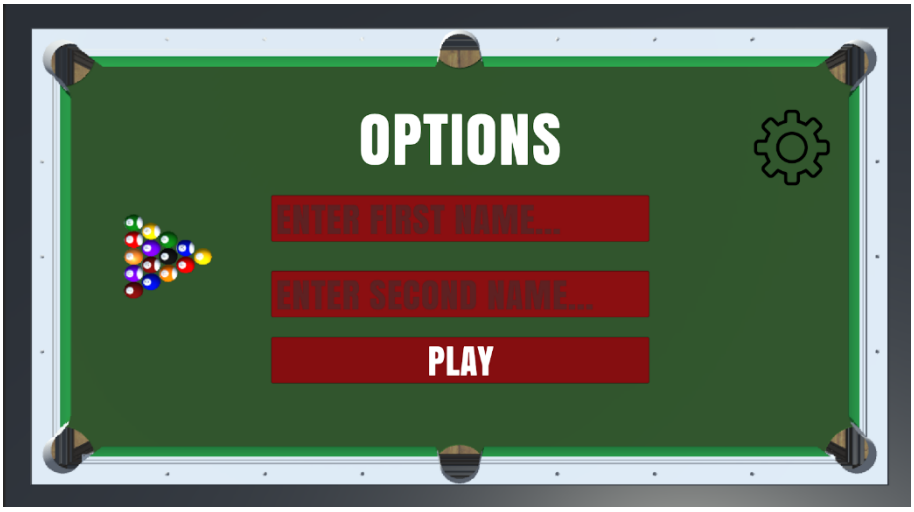
Ziele zu Beginn/ Endergebnis:

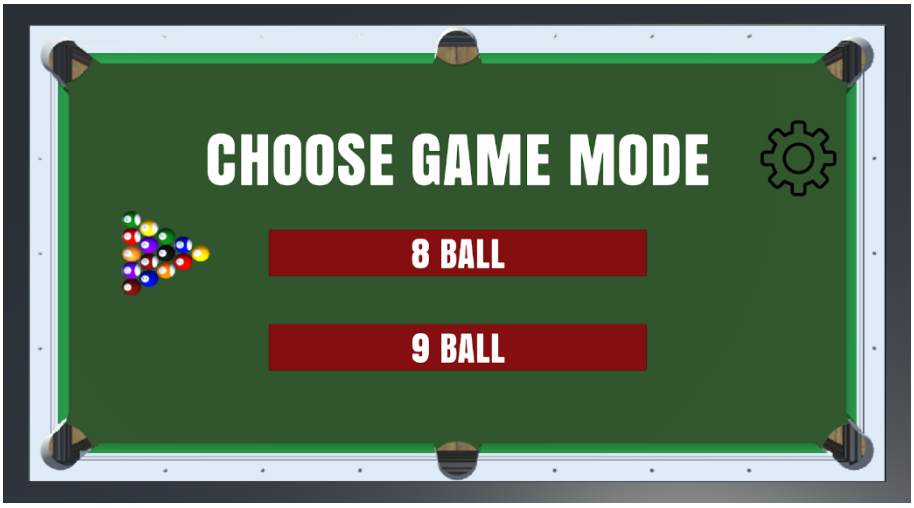
Hauptziel: Benutzbare 3D-Modelle sowie eine funktionierende GUI

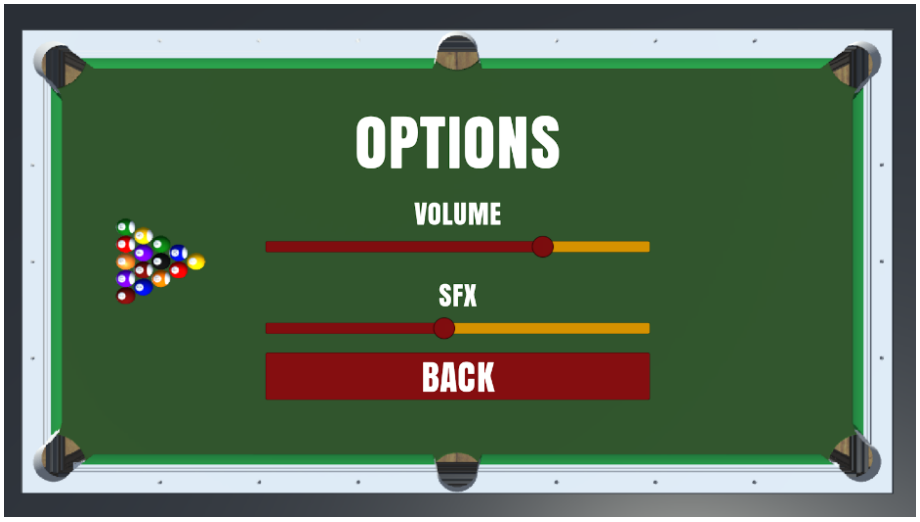
Detailliert alle Ziele:

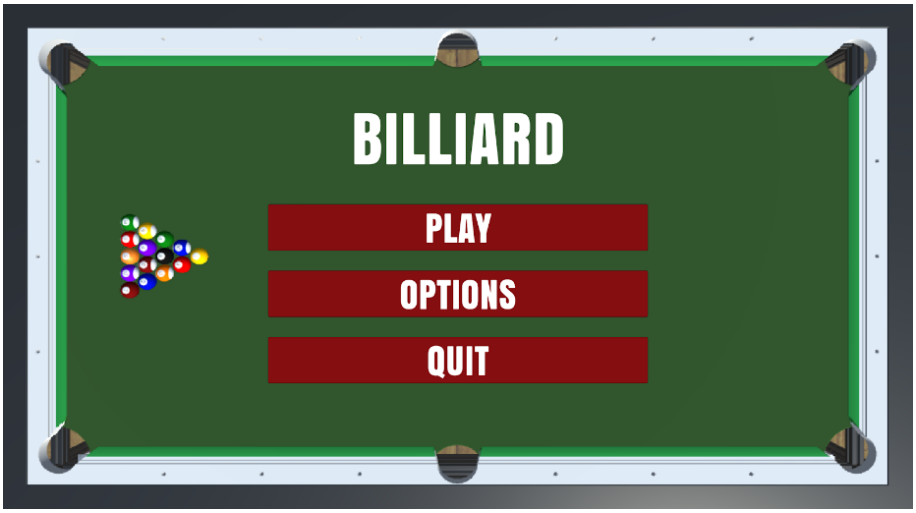
- **3D Modelle:**
  - aus altem Projekt extrahieren und aufbereiten
  - in realistischer Größe einfügen
  - benötigten Modellen mittels Blender erstellen
  - Finden geeigneter Texturen und Beleuchtung einstellen
- **GUI**:
  - Designen und implementieren
  - Funktionalitäten auf Basis der Bedürfnisse der Teams umsetzen
- **Musik**:
  - Musik und Effekte finden und einbinden

Es wurden alle Ziele erreicht und das Endergebnis sieht aus wie folgt:




CHOOSE GAME MODE
8 BALL
9 BALL


OPTIONS
VOLUME
SFX
BACK


BILLIARD
PLAY
OPTIONS
QUIT

Beispiel für erarbeiteten Modelle und Texturen:

Teamarbeit und Kommunikation:

Mittel zur Kommunikation innerhalb des Teams:

- WhatsApp
  - Zur schnellen Kommunikation und Terminfindung
- Discord
  - Zur Organisation mit anderen Teams aber auch um wichtige Meilensteine festzuhalten
- Gitlab
  - Zur Übersicht der geleisteten Meilensteine aber auch um Issues besser im Blick zu haben

Während eines Sprints:

- Zielsetzung für den Sprint innerhalb des Teams
  - Was ist momentan wichtig?
  - Was brauchen die anderen Teams?
  - Was schaffen wir in zwei Wochen?
- Aufgabenverteilung innerhalb des Teams
- Kurzes Review nach einer Woche, gegebenenfalls Ziele anpassen
  - Was wurde bis jetzt geschafft?
  - Was fehlt und ist es noch machbar?
- Tag vor Stand-Up, letztes Update

Der Weg des Teams:

- **1. Sprint:**
  Im ersten Sprint haben wir uns mit den Texturen und Modellen des vorherigen Teams auseinandergesetzt. Wir haben die besten Modelle übernommen und eigene noch über Blender erstellt. Bei den Modellen handelt es sich um den Tisch, die Kugeln sowie den Queue. Dann haben wir uns Texturen und Materialien gesucht die wir auf die Modelle legen können. Dies hat alles einwandfrei funktioniert und wir haben uns als Team gemeinsam in Unity sowie ein wenig in Blender eingearbeitet.

- **2. Sprint:**
  Als nächstes Ziel wollten wir eine GUI entwickeln die zu dem Spiel passt und ansprechend aussieht. Dabei hat sich jeder im Team ein wenig Gedanken gemacht welches Design gut aussehen würde. Danach haben wir im Team entschieden welches Design umgesetzt werden soll. Sobald die GUI implementiert wurde haben wir den Knöpfen Funktionen gegeben. Die Aufgabenverteilung sowie die Bearbeitung der Aufgaben verliefen gut und flüssig.

- **3. Sprint:**
  Für den dritten Sprint wollten wir einerseits Sounds finden um diese als Sound-Effekt einbauen zu können, andererseits wollten wir Musik finden die als Hintergrundmusik dienen könnte. Die Sound-Effekte dienen dazu das Spiel etwas lebendiger wirken zu lassen aber auch um dem Spieler beim Drücken der Knöpfe mehr "Spiel Feeling" zu geben.
  Um das umsetzen zu können mussten erst gute kostenlose Sounds und Musik gefunden werden. Das hat etwas länger gedauert und es könnten nicht alle Sounds direkt eingebaut werden.

- **4. Sprint:**
  Im vierten Sprint haben wir die fehlenden Sounds eingefügt und dazu einen Volumenregler erstellt und angebunden. Dieser soll dem Spieler im Options Menü die Möglichkeit geben die Hintergrundmusik und die Sound-Effekte getrennt voneinander einstellen zu können. Dazu haben wir die Größe des Tisches vereinheitlicht für alle Teams.

- **5. Sprint:**
  Für den letzten Sprint haben wir verschiedene Anforderungen anderer Teams erfüllt, wie zum Beispiel einen Button um die Netzwerkanbindung zu starten bzw. zu stoppen. Des Weiteren haben wir unseren erarbeiteten GUI Stand auf das Team Game übertragen damit diese mit unserer GUI arbeiten können. Zuletzt haben wir die GUI auf alle Bildschirmgrößen angepasst. Wir haben uns im Team immer sehr gut verstanden und kommuniziert.

Bugs die noch auftreten:

- Beim Wechsel einer Szene wird die Musik neu gestartet

Vorausblick:

- Erweitern des Sound Designs
  - Sound beim Einlochen einer Kugel
  - Sound bei Gewinn, Foul, usw.
  - Bessere Sounds und Musik finden
- Visuelle Repräsentation von Ereignissen
  - Blaues Licht beim Einlochen einer Kugel
  - Animation bei Gewinn
- Erweitern der GUI:
  - für weitere Spiel Modi
  - Darstellung einer Bestenliste

Projekt Dokumentation: Team Simulator

Ziele zu Beginn

Kernziel:

- Simulation aller Elemente eines virtuellen Billard-Tisches

Ziele detailliert:

- Objekteinstellungen setzen (Physics)
- Schussstärke und Löcher implementieren
- Pfeil mit Winkelberechnung
- Kö an die weiße Kugel anheften
- Umpositionieren der Kugeln durch Rechtsklick
- Schönheits-Details der Spielumgebung

Es wurden alle Ziele, bis auf den Pfeil mit der dazugehörigen Winkelberechnung erreicht.

Kommunikation und Ausgangssituation

Kommunikation:

- Treffen mind. zwei Mal die Woche
- Keine Aufteilung der Aufgaben; alle Ziele des Sprints wurden zusammen gelöst
- Sehr gute Zusammenarbeit
- Offene Kommunikation über Probleme und Terminabsprache

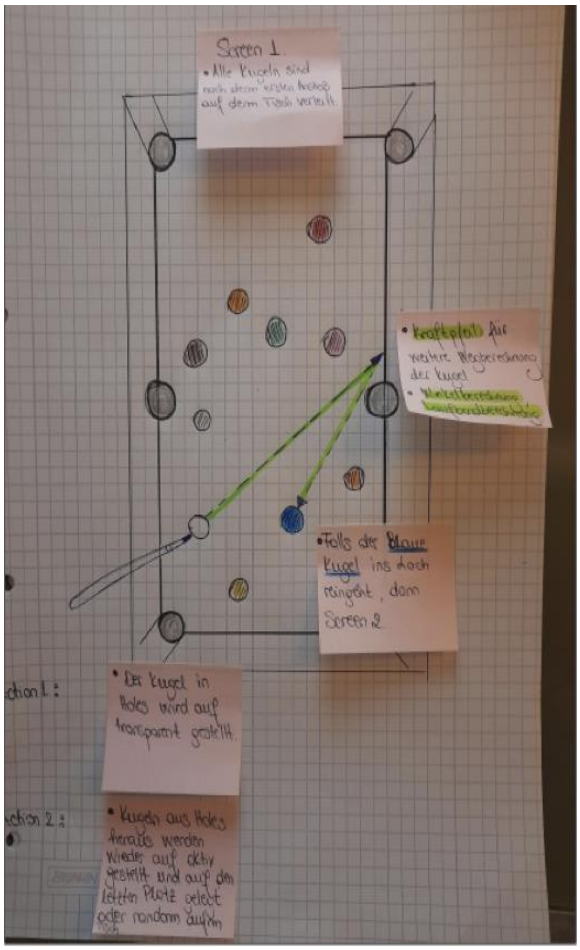
Ausgangssituation:

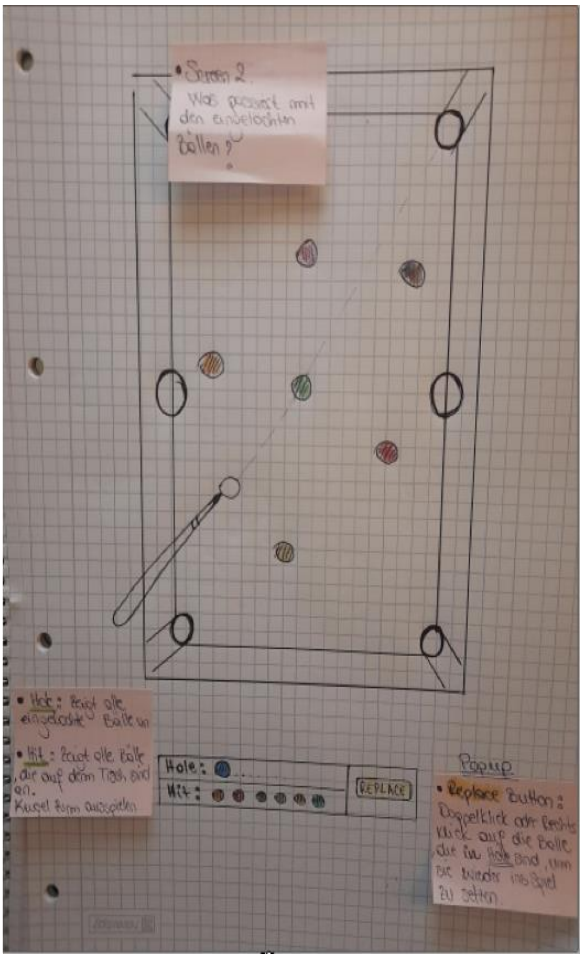
- Vorhandene Projektstruktur untersucht
- In den vorhandenen alten Code einarbeiten
- Altes Projekt als Inspiration verwendet

Der Weg des Teams

- **1.Sprint:**

  Wir haben uns mit dem vorhandenen Code vom vorherigen Projektteam beschäftigt und ihn versucht zu refactoren. Da das vorhandene Projekt aber leider nicht zu unserer neuen Projektstruktur passte, entschieden wir uns ihn zu verwerfen.
  Anschließend haben wir uns Gedanken über den ‚neuen Simulator' gemacht.
  Dabei ist uns aufgefallen, dass wir alle sehr unterschiedliche Vorstellungen haben. Aus diesem Grund hat jeder für sich einen eigenen Paper-Prototypen erstellt (siehe Abbildungen unten) und am Ende haben wir uns zusammen auf einen geeinigt und der gesamten Gruppe beim Sprint Review vorgestellt. Des Weiteren hatten wir ein Gespräch mit Hr. Frömmer um unsere Ziele für dieses Projekt insgesamt besser zu verstehen bzw. herauszufinden.





- **2. Sprint:**

  Nachdem uns klar wurde wie der neue Simulator auszusehen hat, haben wir die Objekteinstellungen gesetzt. Dazu gehörte die Masse von den Kugeln, dynamic friction, static friction und die Schwerkraft. Die Werte haben wir durch ausprobieren erhalten. Die physikalischen Einstellungen, dynamic friction und static friction, beschreiben die Reibung von kollidierenden Objekten.
  Des Weiteren haben wir uns mit der Schussstärke beschäftigt. Im alten Projekt wurde dies mit Pull-and-Release realisiert, wir haben uns jedoch für addForce entschieden. Hierbei drückt man, mit der linken Maustaste, auf die Position die man haben möchte auf dem Tisch um zu schießen. Je länger man die linke Maustaste gedrückt hält, desto stärker ist der Schuss.

- **3. Sprint:**

  Im weiteren Verlauf haben wir eine Box erstellt und die sechs Taschen vom Billardtisch implementiert. Wenn eine Kugel in die Tasche versenkt wird, ist sie für eine Sekunde noch zu sehen, dies haben wir für den realistischen Effekt eingebaut. Nach dieser Sekunde wird die Kugel in die Box ‚teleportiert'. Dies gilt jedoch nicht für die weiße Kugel, sie wird an ihre Anfangsposition zurückgesetzt. Außerdem wird durch betätigen der Leertaste das komplette Spiel resettet.

- **4.Sprint:**

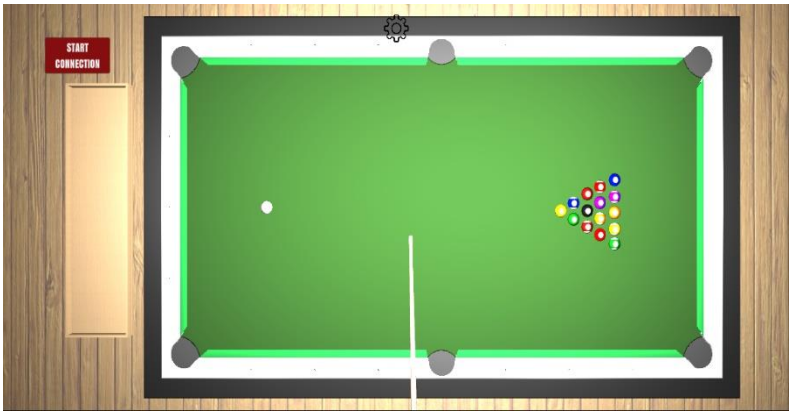
  Da der Billardtisch, den wir bisher benutzt haben, nicht die Originalmaße eines Billardtisches hatte, wurde dies von Team-Graphics korrigiert.
  Hiernach haben sich die Kugeln anders verhalten, weshalb wir die Physikalischen Einstellungen sowie die Skripte nochmal durchgegangen sind und angepasst haben. Für den

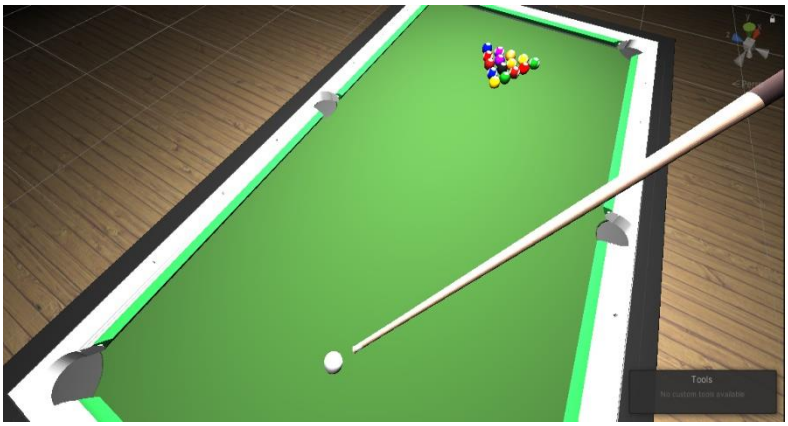
neuen Tisch mussten wir die Box auch erneut erstellen. Nach dieser Änderung haben wir eine Drag-and-Drop Funktion für die Kugeln die in der Box sind implementiert, damit wir die Kugeln zurück auf den Billardtisch umpositionieren können. Die Kugel die man zurück auf dem Tisch haben möchte, klickt man mit der rechten Maustaste an und lässt erst los, wenn man die gewollte Position erreicht hat. Zusätzlich haben wir die Kugeln so ausgerichtet, sodass man die Zahlen auf den Kugeln am Anfang des Spiels sieht.

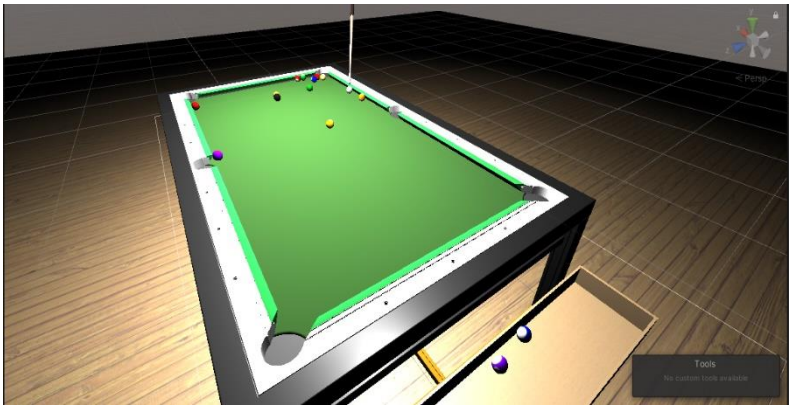
- **5.Sprint:**

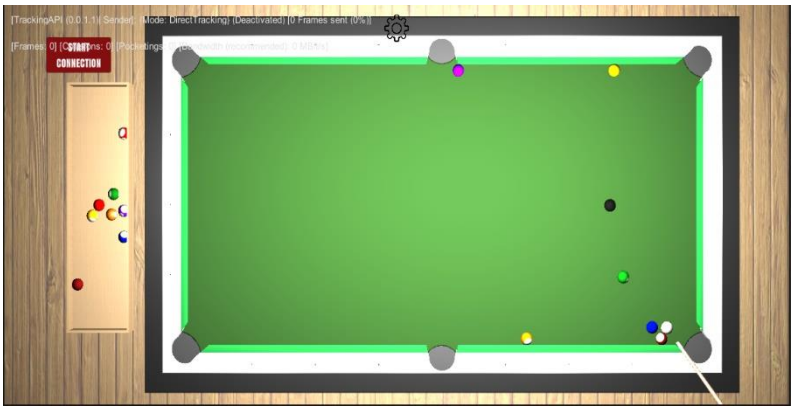
  Im letzten Sprint haben wir neue Lichtquellen hinzugefügt, die Tracking API eingebunden und nochmals mit den Physics gespielt, da das Rollverhalten uns immer noch nicht gefiel. Zudem ist uns ein Bug aufgefallen, dass wenn man zu stark schießt, dass die weiße Kugel durch die anderen Kugeln rollt. Dies haben wir behoben, in dem wir die Einstellungen bei Collision Detection verändert haben. Collision Detection stellt sicher, dass sich schnell bewegende Körper mit Objekten kollidieren, anstatt diese Objekte zu passieren oder zu tunneln. Zu guter Letzt haben wir die Queue an die weiße Kugel angeheftet. Die Queue ist nur sichtbar, wenn die weiße Kugel stillsteht, das heißt sobald man schießt wird der Queue unsichtbar.

Endergebnis









Physikalische Einstellungen

- Im Rigidbody der Kugeln: Collision Detection - alle Kugeln außer weiße Kugel auf Continious Dynamic
  —> damit die weiße Kugel nicht durch die anderen Kugeln durchrollt, wenn man zu stark stößt
- In den Einstellungen der Assets/GameObjects/Material/ tableBoarder: Dynamic Friction = 0.475; Static Friction = 0.475;
  —> durch ausprobieren mit den vorhanden Physics, damit sie sich nicht wie durch „Gelee“ bewegen
- Alle Kugeln: Mass= 0.02; Drag = 0; Angular Drag =1
  —> Berechnung der Masse: Verhältnis von realer Masse und Schwerkraft zu Unity Einheiten
- Project Settings → Physics → Gravity = -9,81
  —> Default von Unity

Hinweise zum Code

- Für Weltkoordinaten haben wir leere Objekte an der Stelle erstellt wo wir wollten, dass das jeweilige Objekt hin teleportiert wird. Von diesem Objekt konnte man die Koordinaten dann ablesen.

Bugs die noch auftreten

- Beim Anstoß rollen die Kugeln nur in die Richtung, in die auch geschossen wird. Rollverhalten müsste korrigiert werden.
- Der Queue verschwindet manchmal unter dem Tisch, wenn man ihn in einem ganz bestimmten Winkel hält.

Vorausblick

- Winkel Vorausberechnung für den Aufprall der weißen Kugel mit der Bande und den anderen Kugeln.
- Ladebalken für die Schussstärke.
- „Bessere“ physikalische Einstellungen für die Kugeln, damit sie „schöner“ rollen.

Projekt Dokumentation Team Tracking

Ziele zu Beginn

- Datenübertragung zwischen Simulator/Billardtisch und der Spiellogik
- Tracking Data Konzeption
- Tracking des Simulators/Billardtisch

Kommunikation

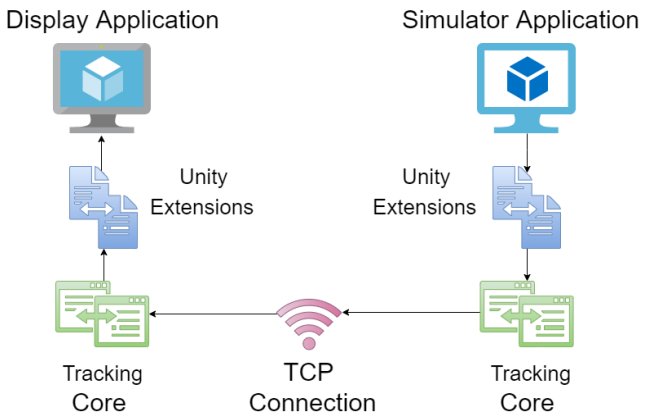
- Team aus zwei Leuten
- Kommunikation lief komplett über Discord
- Da nur zwei Leute kaum Aufgabenteilung, im Normalfall wurde einfach gemeinsam an der derzeitigen Aufgabe gearbeitet

Der Weg des Teams

1. Erarbeitung des Konzepts

   Wir haben zu Beginn die alte Tracking Software, aufgrund von unzureichender Dokumentation und damit das neue Gesamtprojekt einheitlich in C# geschrieben ist, verworfen.
   Unser neues Konzept wird in *Abbildung 1* dargestellt. Ziel war es das es möglichst egal ist aus welcher Quelle die API ihre Daten bekommt. Eine Instanz der API (Tracking Core) soll die Daten erhalten, diese serialisieren und komprimieren und daraufhin über TCP an eine zweite Instanz der API schicken. Diese soll die Daten entkomprimieren und deserialisieren und daraufhin im richtigen Format weiterleiten. Des Weiteren werden Unity Extensions geschrieben welche die Schnittstelle zwischen dem Simulator / der Spiellogik und der API bilden.

   Abbildung 1:



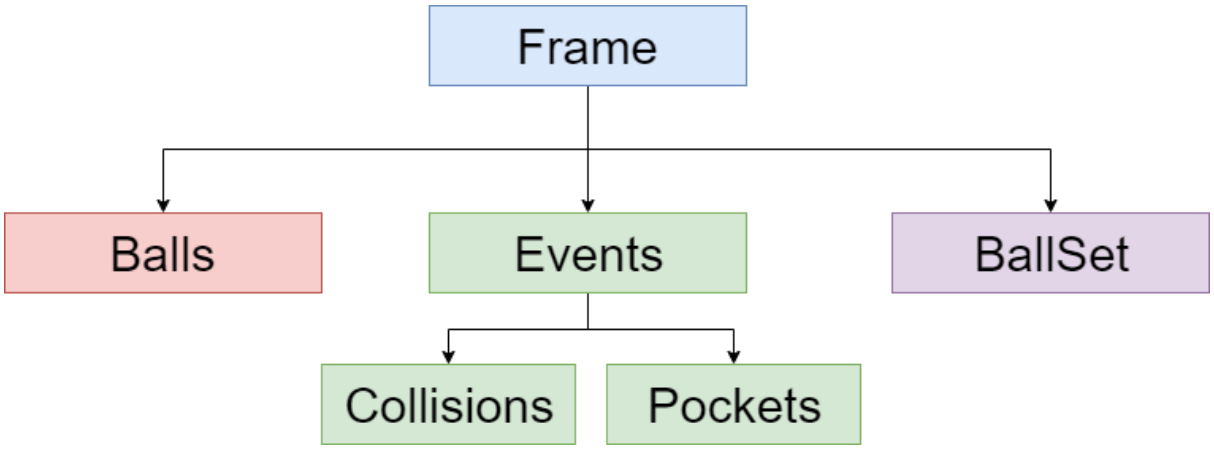
2. Die Tracking API

Als ersten haben wir den Code der Tracking API geschrieben.

Dies bestand aus der Frame Klasse welche das Datengerüst stellt für die Information enthält, welche die Spiellogik benötigt. Ein Frame enthält Daten über:

- welche Bälle im Spiel sind
- die Position, Rotation und Geschwindigkeit der einzelnen Bälle
- Kollisionen zwischen den Bällen
- welche Bälle wo eingelocht wurden.

Abbildung 2:



Des Weiteren wurden Funktionen zur Umwandlung der Daten aus und in Unity konforme Konstrukte geschrieben. Und ein Klasse welche als Gerüst für die später kommende Datenübertragung dienen sollte. Das nächste Ziel war es eine Schnittstelle für das Game Team zu entwickeln damit diese lokal anfangen konnten ihre Spielregeln zu testen.

3. Schnittstelle für die Spiellogik: Tracking Receiver

Der Tracking Receiver erhält Daten über die API, welche er dann in die Unity Anwendung des Team Game einbindet. Da zu diesem Zeitpunkt noch keine Daten übertragen werden konnten wurde ein lokales Tracking erstellt. Hierbei wurden die Spieldaten direkt aus der Anwendung des Team Game verwendet und in die Datengerüste der API umgewandelt. Dies ging dann wieder zum Receiver welcher die Daten in dem Format vorliegen hatte in welchem er auch später die Übertragenen Daten erhalten würde. Unser nächster Schritt war die Schnittstelle für das Team Simulator.

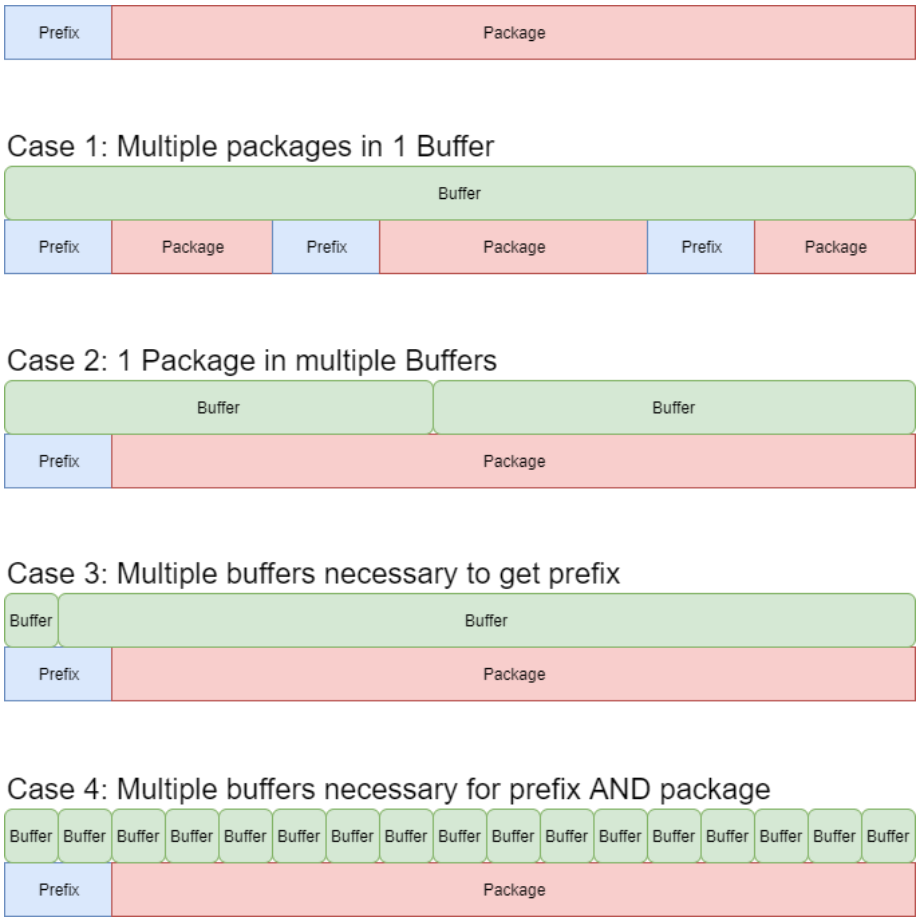
4. Schnittstelle für den Simulator: Tracking Sender

Der Tracking Sender nimmt sich die benötigten Daten direkt aus der Unity Anwendung des Team Simulators und wandelt diese in das zu verschickende Format um. Hierfür nutzt er die vorher erarbeiteten Funktionalitäten der Tracking API. Als nächster Schritt kam die Übertragung der Daten.

5. Datenübertragung und Änderungen an den Schnittstellen

Nun wurden die zuvor in der Tracking API erstellten Gerüste gefüllt. Als Protokoll für die Übertragung wurde TCP gewählt. Dies erschien uns die logischste Wahl, da wir einen konstanten Stream zwischen Sender und Empfänger haben und Daten korrekt und in der richtigen Reihenfolge ankommen sollten. Des Weiteren läuft die reelle Anwendung nur in einem Heimnetzwerk, insofern sollte die Geschwindigkeit von TCP mehr als ausreichend sein. Damit TCP funktioniert mussten eine Klasse schreiben, welche erreicht, dass die Framedaten (unsere Packages) beim Empfänger richtig aus dem Stream ausgelesen werden. Unsere Klasse Package Extractor sorgt dafür, dass Packages die im Buffer des Empfängers liegen richtig aus diesem ausgelesen werden und im Zweifelsfall wieder korrekt zusammengesetzt werden. *Abbildung 3* zeigt die Fälle die der Package Extractor behebt.

Abbildung 3:



Zuletzt wurden die Funktionen zur Serialisierung und Komprimierung und Deserialisierung und Dekomprimierung eingefügt.
Die Schnittstellen wurden angepasst damit die nun übertragenen Daten benutzt werden konnten. Über den Prozess der Einbindung der Datenübertragung wurden mehrere Tests geschrieben um zu gewährleisten, dass das System stabil ist.

Endergebnis

Das Endergebnis unseres Teams ist eine API, welche einfach zu bedienen und zu erweitern ist. Umgesetzt wurden die Kern API und die Schnittstellen zwischen ihr und den jeweiligen Unity Anbindungen. Von der ursprünglichen Zielsetzung fehlt das Tracking. Grund hierfür war die geringe Größe unseres Teams. Im Laufe des Projekts haben wir unseren Fokus mehr auf die Stabilität und Integrität der Übertragung gesetzt, damit ein insgesamt rundes Gesamtprojekt entsteht.

Hinweise

Wie man die API aufsetzt findet ihr in "setup-guideline.md".
Für alles weitere am besten in die Code Doku schauen.

Bugs

Unseres Wissens ist die Tracking API und die Schnittstellen zu Unity Bug frei.

Für die Zukunft

- Tracking anhand von Bilddaten
    - die Tracking API enthält zurzeit nur einen Platzhalter, hier kann man also ansetzen. Vorschlag von uns: nutzt keine AI oder neuronalen Netze dafür das Team vor uns hat das versucht und es sah nicht schön aus. Am besten auf das Verlassen was man aus Grafische Datenverarbeitung kennt.
- Performance improvements
- DNS in die Software integrieren

Projekt Dokumentation: Team Game

Ziele

- **Hauptziel:** Implementierung der Spiellogik des "Multimedia Billard Table"
- **Ziele detailliert:**
  - Allgemeine Spiellogik des Spiels
  - 8-Ball-Pool
  - 9-Ball-Pool
  - Regelverstoß
  - Konsequenzen
- **Es wurden alle gesetzten Ziele erreicht.**

Kommunikation

- Kommunikation erfolgte über Discord & WhatsApp
- Sowohl gemeinsames Arbeiten als auch Einzeln
- Interne Kommunikation lief gut
- Vereinzelte Missverständnisse zwischen den Teams

Ausgangssituation

- Versucht den vorhandenen Code zu verstehen/erweitern
- Was sind unsere Aufgaben?
- Wofür sind wir zuständig? (-> ganz wichtig vorab abzuklären, damit keine Missverständnisse entstehen und somit zu Zeitverlust führt)

Der Weg des Teams

- **1.Sprint:**

  Unsere Aufgabe als das Game-Team ist es die Logik und Struktur des ‚Multimedia Billard'-Spiels zu implementieren. Da wir vom vorherigen Projektteam einen vorgefertigten Code zur Verfügung gestellt bekommen haben, mussten wir uns erst mit diesem auseinandersetzen. Wir planten zuerst wie wir das Projekt mit unserem Teil am besten umsetzen und schätzten die Aufwände der jeweiligen Sprints ein. Ziel dieses Sprints war es, den ein Refactoring des vorhandenen Codes vorzunehmen und die benötigten IDE (Unity, VS, Git ...) zu installieren und vorzubereiten. Zudem bemerkten wir schon anfangs, dass der Code möglicherweise schwere Bugs enthielt, da das Spiel in Unity sehr oft abstürzte/einfrierte. Wir detektierten die Bugs in dem zugeteilten Code und protokollierten diese um sie im zweiten Sprint zu fixen. Uns ist auch aufgefallen, dass alle verschiedene ‚Versionen' haben was die Helligkeit, Anzeigen des Kö's etc. betrifft. Wir teilten den Code in unserem Team auf, indem jeder eine Funktion/Klasse bearbeitet hat. Ausgenommen einer Funktion (die relativ ausgiebig war) haben wir es geschafft den Code zu verbessern und verständlicher zu schreiben. Es ist geplant im nächsten Sprint die Bugs zu fixen und die bisherigen Regelverstöße zu protokollieren und ggf. anzupassen

- **2.Sprint:**

  Bezüglich des Projekts hielten wir Rücksprache mit den anderen Gruppen. Dabei kam raus, dass wir als Game-Team einen eigenen Code haben (unabhängig von anderen Teams z. B. Simulator) und diesen bearbeiteten. Durch diese Erkenntnis haben wir uns entschieden von vorne anzufangen und einen neuen verständlichen Code zu schreiben, der leicht erweiterbar sein soll. Wir beschlossen auch den Billardtisch und die Bälle in Unity neu aufzusetzen. Eine Debugfunktion um die Bälle zuspielen haben wir ebenfalls implementiert und dazu die Kollisionserkennung mit Bällen und Banden hinzugefügt um unsere Spiellogik für den späteren Zeitpunkt testen zu können. Somit haben wir unseren Planungen geschafft.

- **3.Sprint:**

  Wir haben die Klassenstruktur ausführlich geplant und festgelegt. Dazu haben wir ein UML-Diagramm festgelegt und angefangen die benötigten Klassen zu implementieren. Da es viel Zeit in Anspruch nehmen wird die gesamten 8-Ball Pool Regeln zu programmieren, werden sich unsere Sprintziele über mehrere Sprints ziehen um es komplett zu fertigen. Wir sind momentan dabei die allgemeine Spiellogik zu implementieren und die 8-Ball Pool Regeln umzusetzen.

- **4.Sprint:**

  Durch die interne Aufgabenverteilung in unserem Team haben wir es geschafft das Regelsystem und die allgemeine Spiellogik aufzubauen und zu implementieren. Soweit stehen die Regeln für 8-Ball Pool (siehe 'Foulregeln 8-Ball Pool') bis auf ein paar Erweiterungen/Verbesserungen der Methoden, die wir uns für den nächsten Sprint vorgenommen haben um diese zu vervollständigen. Falls noch Zeit übrigbleibt, würden wir mit den 9-Ball Pool Regeln anfangen aber dies haben wir uns nicht als Ziel für den nächsten Sprint angesetzt, weil es doch sehr kritisch mit dem Zeitmanagement aussieht.

- **5. Sprint:**

  Der Spielmodus 8-Ball-Pool ist nun fertig implementiert. Alle Klassen und Methoden sind vervollständigt und tun das was sie sollen. Wir haben den Simulator-Tisch und die UI vom Team Graphics eingebunden. Diesbezüglich implementieren wir die Schnittstelle. Zudem haben wir es geschafft den 9-Ball-Modus zu implementieren auch wenn dies nicht zu unseren „Haupt"-Zielen aufgrund anfänglicher Komplikationen gehörte. Unser Code ist leicht erweiterbar für die fehlenden Modi (Bandenregeln, Turnier Regeln etc.).

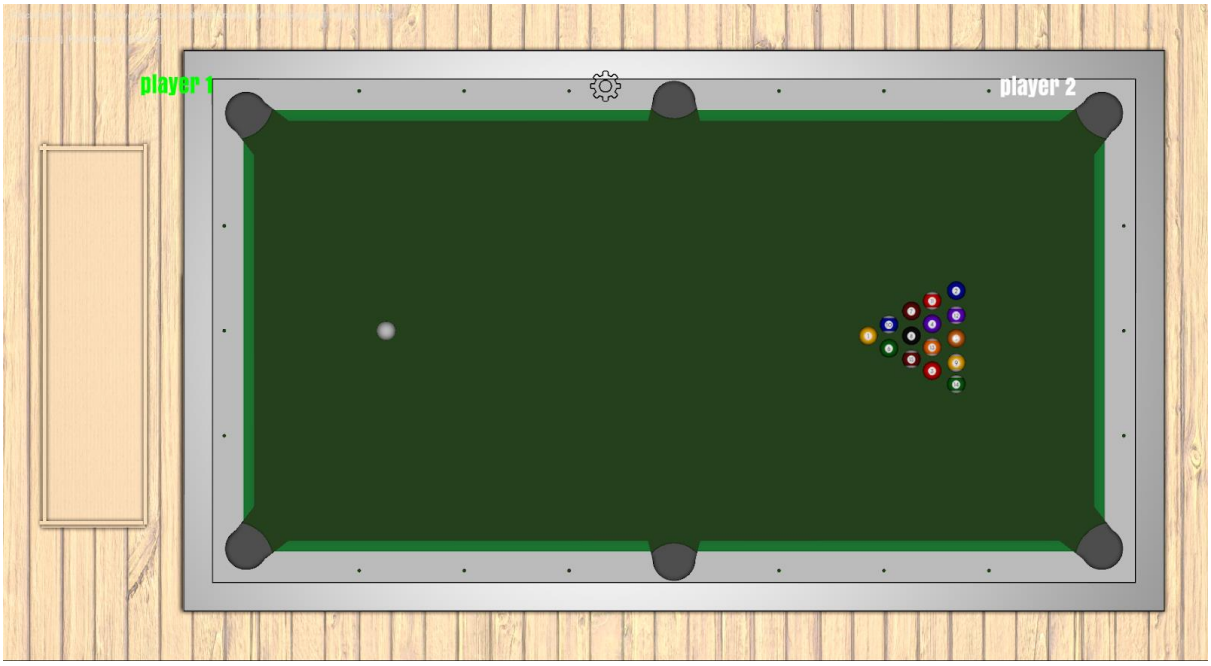
Bugs die noch auftreten

- Wenn beim 8-Ball-Pool die letzte Kugel eingelocht ist, ist man berechtigt die schwarze Kugel anspielen zu können. Jedoch wird ein Wrong-Touch-Foul ausgelöst.
  - Es gibt also bei der Wrong-Touch-Rule einen Bug. Es werden bei GetRemainingBalls vermutlich nicht korrekt die übrig gebliebenen Bälle zurückgegeben.

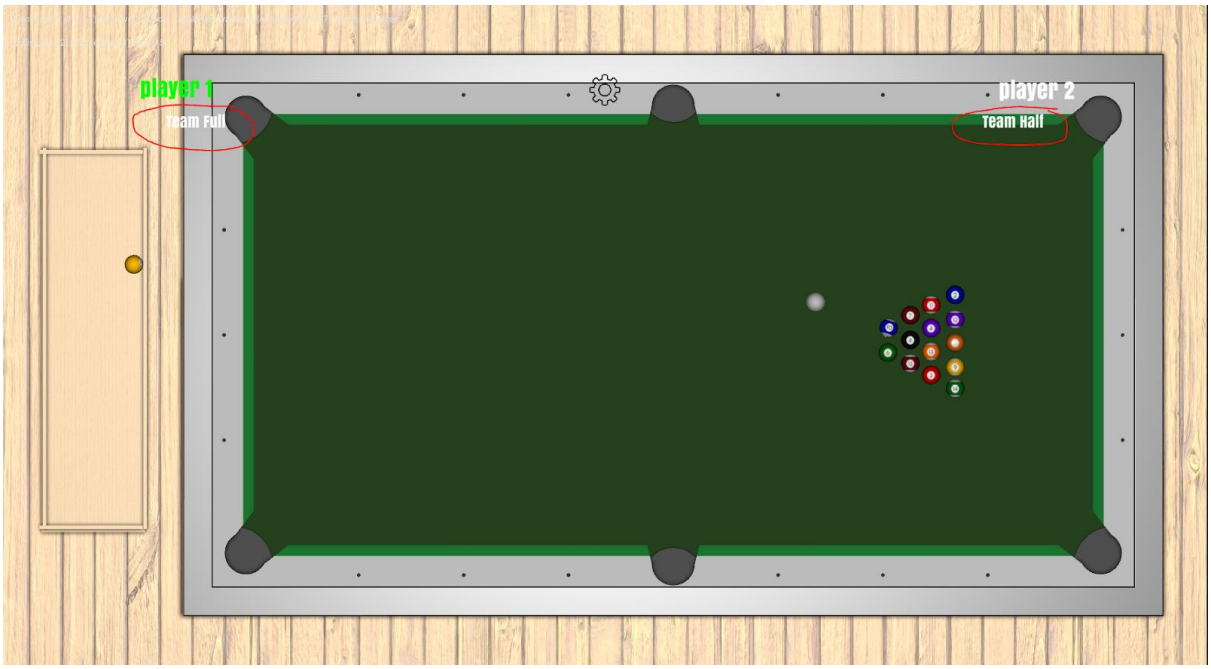
Endergebnis

Beispiel

Das ist 8-Ball-Pool zu Beginn. Der Spieler, der spielen darf, wird grün markiert (player1).



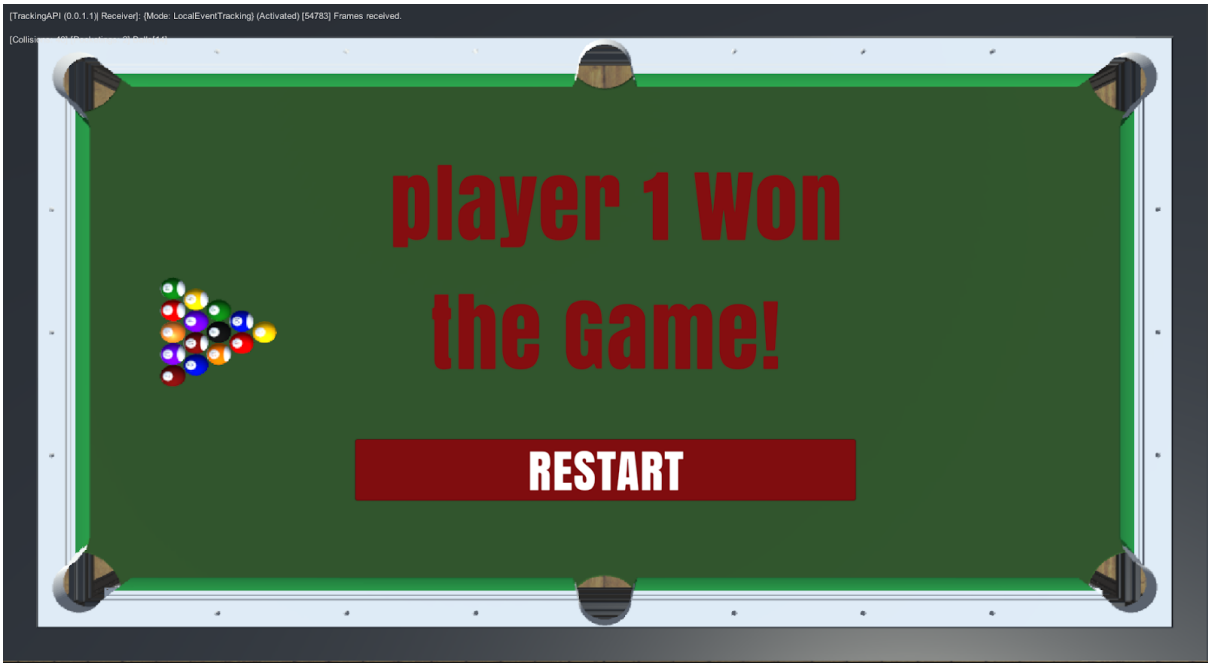
Nachdem eine Kugel eingelocht wurde



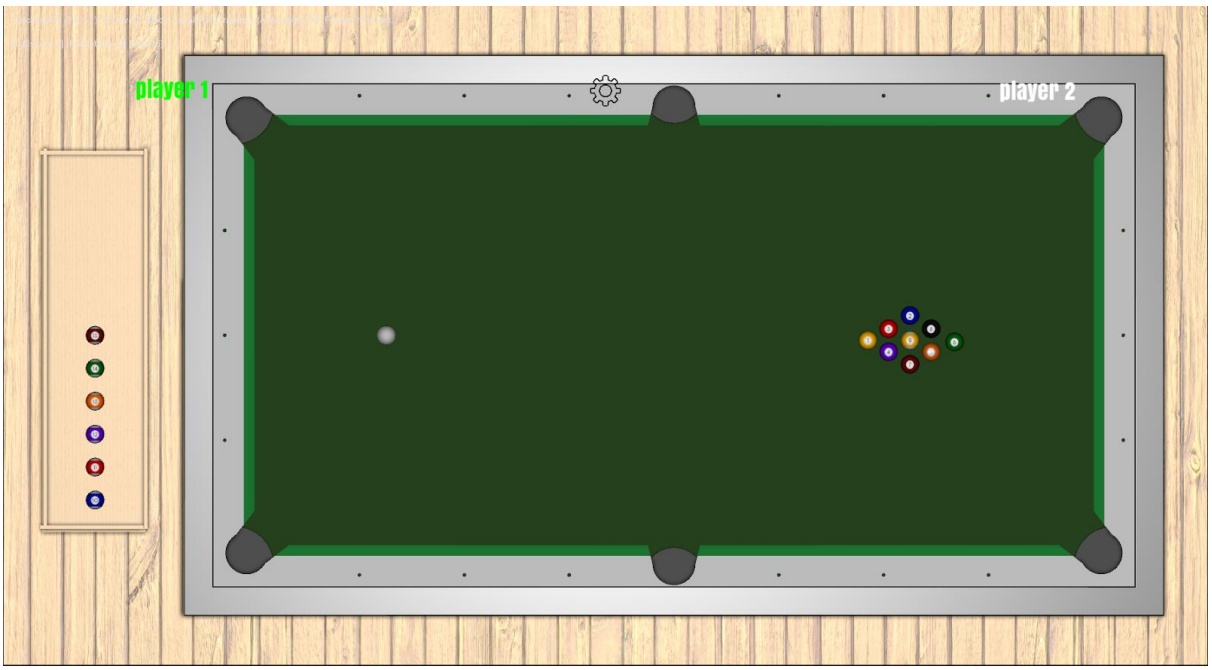
Wenn keine Kugel erwischt wurde und Spielerwechsel.

Jetzt hat player2 unberechtigt die schwarze Kugel eingelocht.

player1 gewinnt.

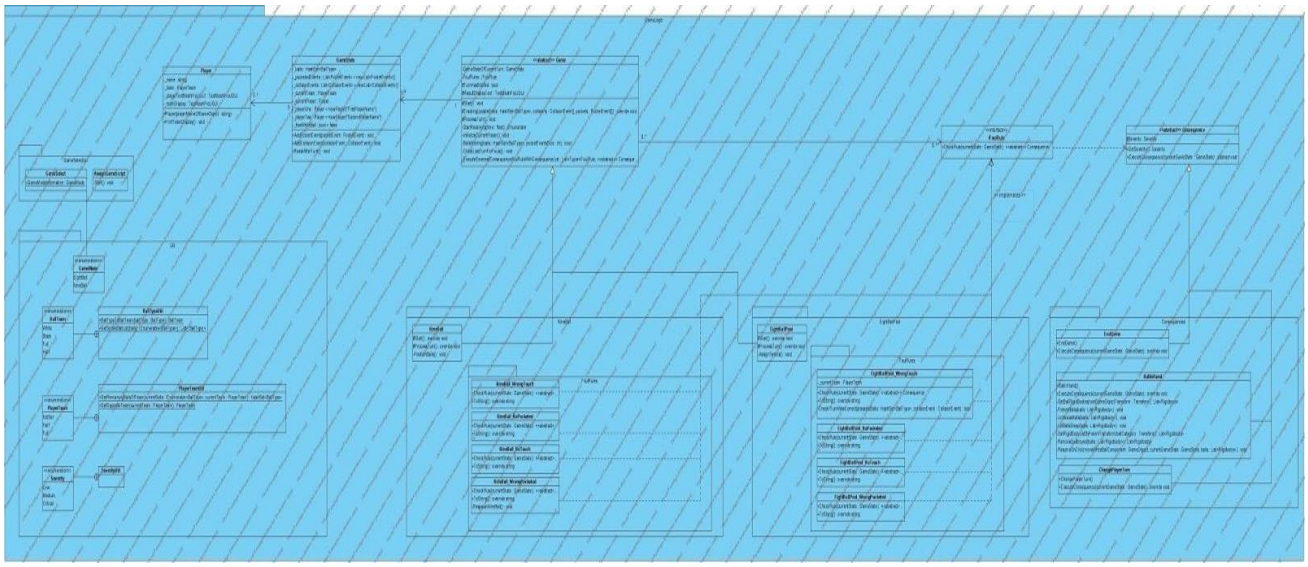


9-Ball-Pool -> grün markierter Spieler ist am Zug.



Player1 hat die Kugel 9 eingelocht, obwohl er die Kugel 1 nicht zuerst angespielt hat. Spielerwechsel und Kugel 9 wird in die Mitte platziert.

UML-Klassendiagramm zur Codestruktur



Vorausblick

- Spiel:
  - Bandenregeln
  - Turnier Regeln
  - Trick-Shot Modus
  - Custom Spielmodus (Mit selbst gewählten Regel Set)

- Code Qualität:
  - Refactoring (Util Klassen durch konkrete Klassen ersetzen usw.)
  - Bug fixen
  - Tests schreiben

6. Schwierigkeiten und deren Lösungen

Probleme	Lösungen
• Unklarheiten zu Beginn des Projekts	• Viel Kommunikation inner- und außerhalb der Teams
• Einarbeiten in C# und Unity ○ Schwierigkeiten bei Vererbung mit Mono Behaviour usw.	• Hilfestellung durch Head-of-Tech • Unity und C# Foren genutzt • Vorlesungsfolien aus anderen Modulen für C# genutzt
• Missverständnisse im Aufbau des Projekts	• Gemeinsames Besprechen mit Product Owner und Scrum Master beteiligter Gruppen
• Missverständnisse bei Besprechungen mit anderen Teams	• Interne Absprache der Verantwortungen • Hilfestellung durch Product Owner
• Aufwandsschätzungen / Festlegen realistischer Sprint Ziele	• Story Points wurden mit jedem Sprint immer genauer durch Vergleichen der Aufwandsschätzungen aus dem vorherigen Sprint → Planning Poker
• unklare Aufgaben wegen unterschiedlichen Spielvorstellungen [Team Simulator]	• Paper Prototype von jedem Teammitglied erstellt und auf eine Version geeinigt • Hilfestellung durch Hr. Frömmer
• Welche Regeln sollen wir benutzen? [Team Game] ○ Können wir gewisse Regeln überhaupt umsetzen?	• Gemeinsame Überlegungen zur möglichen Umsetzung der 8-Ball Regeln, später 9-Ball • Internetrecherche • 8 Ball Pool-App als Inspiration genutzt
• Entwurf der Code Architektur [Team Game] ○ oft neu überdenken	• wurde mit dem Fortschreiten des Projektes immer klarer und wurde so immer wieder an den Stand des Projektes angepasst
• Objekteinstellungen realitätsnah umsetzen [Team Simulator]	• Hilfestellung durch Head-of-Tech • Sehr viel recherchieren in YouTube, Unity Foren und Google
• Nicht genügend „Manpower“ im Team Tracking um Tracking Algorithmus fertigzustellen [Team Tracking]	• Konzentration auf Funktionalität des Gesamtprojektes → Umstellung auf Exchange API
• Teamgrößen teilweise zu unterschiedlich ○ Tracking Team zu klein für alle Funktionalitäten	• Im nächsten Projekt mehr auf den Aufwand für die jeweiligen Gruppen achten → nicht nur auf die Fähigkeiten der Mitglieder

7. Fazit

Das Hauptziel des Projektes wurde erfolgreich erreicht. Es wurden zwei voneinander unabhängige Anwendungen erstellt, deren Daten über eine Netzwerk-Schnittstelle übertragen und ausgewertet werden.

In dem Projekt haben wir gelernt, dass eine offene und respektvolle Kommunikation, eine gute Projektstruktur und eine gute Projektdokumentation absolut essentiell für den Erfolg sind. Die Projektstruktur und Projektdokumentation sollte dabei immer komplett transparent zugänglich sein für das gesamte Team.
Ohne diese drei genannten großen Hauptthemen kommt es sehr leicht zu Missverständnissen und dies kann zu Problemen führen. Jedoch sollte man sich bewusst sein, dass Probleme und Missverständnisse immer Teil eines jeden Projekts sind, doch können sie durch diese drei „Grundpfeiler“ frühzeitig erkannt und behoben werden.