Softwareprojekt

Inhaltsverzeichnis

Michael Herber Fabian Pechta Alexey Khokhlov
Manuel Wiebe Abdul-Kerim Gerikhanov

Softwarespezifikation

Projekt: Puddle Partners

Einführung

Ein Level basiertes Kooperationsspiel mit Rätsel/Puzzle-Mechaniken. Die Spieler müssen sich durch verschiedene Level kämpfen, es gibt Wetterereignisse sowie feindliche gesinnte NPCs, die die Spieler am Fortschritt behindern. Durch Abschluss aller Level oder dem vorzeitigen Ende (Zeit abgelaufen/Spieler-Tod) wird ein Punktwert ermittelt und am Scoreboard veröffentlicht.

Beschreibung

Geplant ist ein 2D side view Pixel Kooperations Spiel in dem es darum geht, in möglichst kurzer Zeit zu zweit Rätsel und Puzzles zu lösen. Das Spiel wird Level-basiert aufgebaut sein, sodass die Schwierigkeit mit steigendem Level stetig ansteigt. Zu den Besonderheiten unseres Kooperations Spiels zählt:

- Regen mit unterschiedlichen negativen Auswirkungen
- NPCs die die Spieler beim lösen der Rätsel und Puzzles stören
- Ein Regenschirm, um den Wetterbedingungen zu trotzen, das jeweils nur ein Spieler benutzen kann, aber beide darunter geschützt sind
- Damit die Wetterereignisse nicht abgewartet werden wird es eine Zeiteinschränkungen geben, wenn die Zeit abgelaufen ist wird das Level geflutet und das Spiel ist beendet

Das Spiel wird kein explizites Kampfsystem erhalten, dennoch haben wir vor, den Spielern Lebenspunkte zu geben. Die Lebenspunkte werden zum größten Teil durch Wetterbedingungen und Level Ereignisse beeinflusst. Die Rätsel und Puzzles werden so ausgelegt, dass diese nur zu zweit gelöst werden können. Am Ende des Spiels wird ein Punktestand ermittelt und auf einer Bestenliste veröffentlicht.

https://github.com/HSBI-Softwareprojekt/Specification

Anforderungen

Stakeholder
Spieler:
Setzt sich mit der Software auseinander (spielt) und möchte ein fehlerfreies aufregendes Spielerlebnis genießen.

Investoren/Teilhaber:
Sind auf Profit aus und erwarten ein Produkt, das die Spieler kaufen möchten und finanzieren das Produkt.

Funktionale Anforderungen

Rahmenbedingungen/Systemanforderungen
Client:
Betriebssystem: Windows 10 und höher (Mobile?)
Prozessor: Wird zur Projekt endphase ermittelt
Grafikkarte: Wird zur Projekt endphase ermittelt
Arbeitsspeicher: Wird zur Projekt endphase ermittelt
Speicherplatz: Wird zur Projekt endphase ermittelt

DB-Server:
Betriebssystem: Windows 10 und höher, Windows Server 2022 / Ubuntu 24.04 LTS
Prozessor: Wird zur Projekt endphase ermittelt
Arbeitsspeicher: Wird zur Projekt endphase ermittelt
Speicherplatz: Wird zur Projekt endphase ermittelt
Netzwerkanbindung: Abhängig von Spieleranzahl

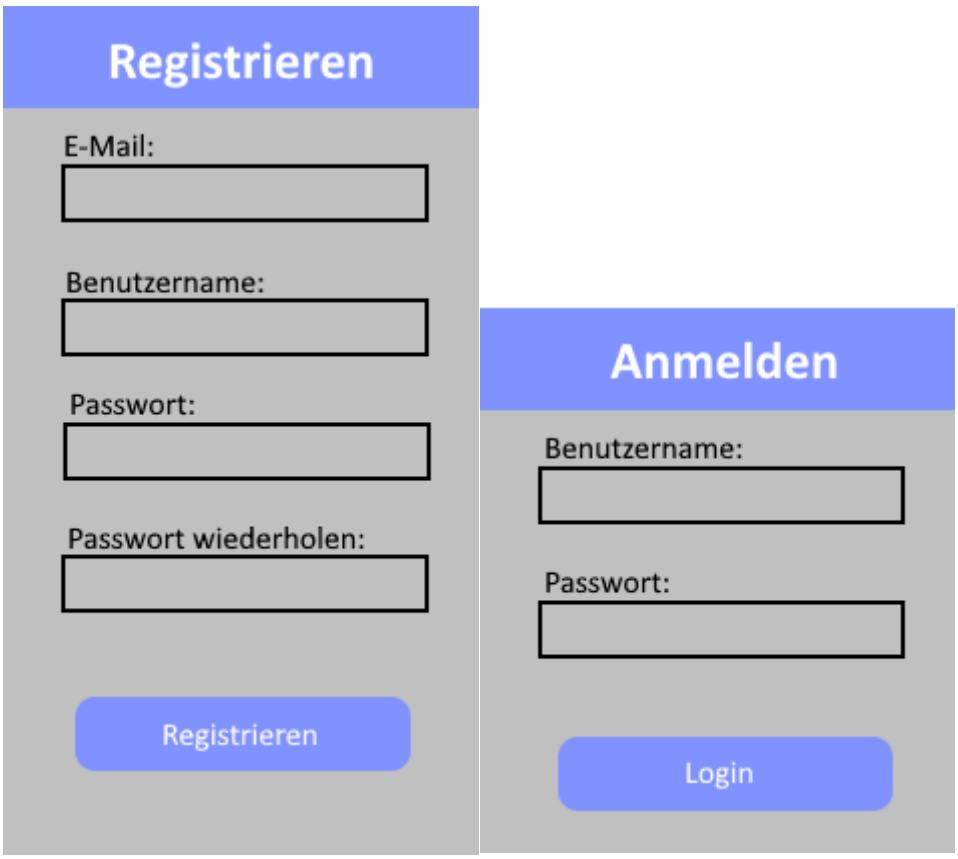
Betriebsbedingungen
Programmiersprache C#
Angeboten als Native App, Desktop Applikation und Web Browser.

Qualitätsmerkmale
|Sehr gut|Gut|Normal|Nicht relevant|

- Fehlertoleranz |-|X|-|-|
- Wiederherstellbarkeit |-|-|-|X|
- Ordnungsmäßigkeit |-|X|-|-|
- Richtigkeit |-|X|-|-|
- Konformität |-|-|-|X|
- Installierbarkeit |X|-|-|-|
- Verständlichkeit |X|-|-|-|
- Erlernbarkeit |X|-|-|-|
- Bedienbarkeit |X|-|-|-|
- Zeitverhalten |-|-|X|-|
- Effizienz |X|-|-|-|

Graphische Benutzerschnittstelle

- Die ganze Anwendung ist eine Benutzerschnittstelle:



https://github.com/HSBI-Softwareprojekt/Specification

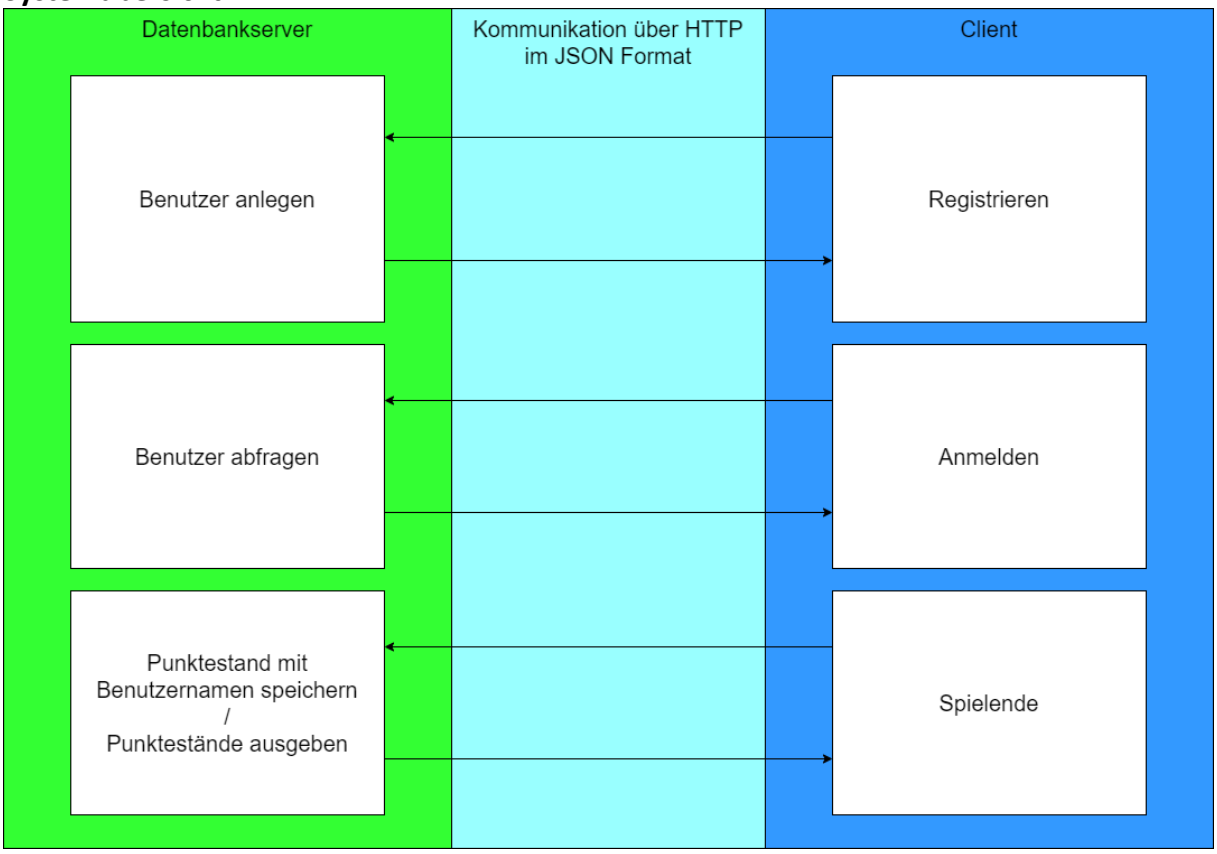
Anforderungen im Detail

User Stories

Als	Möchte ich	Sodass	Akzeptanz
Spieler	Die Möglichkeit haben das Scoreboard zu sehen	Ich mich mit anderen vergleichen kann	Scoreboard anzeigen
Spieler	Meinen Partner sabotieren	Wir, oder nur ich was zu lachen habe	Spieler können sich gegenseitig sabotieren
Spieler	Mit meinem Partner zusammen Herausforderungen meistern	Wir dieses Erfolgserlebnis zusammen teilen können	Level nur in Kooperation abschließbar
Spieler	Einen flüssigen Spielfluss haben	Das Erreichen des Highscores macht	Flüssige Steuerung und Reaktionsschneller Spielfluss
Spieler	Kreative Puzzles lösen	Das Spielen viel Spaß macht	Kreative Ideen für Hindernisse
Spieler	Mit den Puzzles auf die Probe gestellt werden	Ich mich gut fühle, wenn ich die Lösung finde	Anspruchsvolle Level haben (Low Prio)
Spieler	Dass das Spiel flüssig läuft	Das Spielen nicht frustrierend ist	Effiziente Implementierung
Spieler	Mit meiner Umgebung interagieren	Ich Puzzle auf verschiedene Arten lösen kann	Greifmeachnik und Kollisionsbedingungen Einbauen
Spieler	Einen Timer oder alternative Indikatoren für die verbleibende Zeit des Levels	Ich abschätzen kann, ob ich mich beeilen muss, das Rätsel zu lösen	Zeitablauf darstellen
Investor	Ein gutes Spielerlebnis für den Spieler	Das Spiel gut bewertet und öfter gekauft wird	Spieler ist zufrieden
Investor	Möglichst niedrige Systemanforderungen	Potentielle Spieler nicht durch zu hohe Anforderungen am Kauf gehindert werden	Effiziente Implementierung

Technische Beschreibung

Systemübersicht



Softwarearchitektur

Die gesamte Software läuft auf dem System des Benutzers, eine Kommunikation findet beim Registrieren, Anmelden und am Spielende mit einem Datenbankserver für das Hinterlegen der erreichten Punkte und das Anzeigen der Platzierung im Scoreboard statt.

Technologiewahl

Unity (2022.3.26f1) als Entwicklungsplattform, C++, C#, JSON
MySQL als Datenbank Server, PHP als zusätzliche Sicherheitsschicht

Schnittstellen

Intern:

- UnityEngine.UI, um das User Interface zu verwalten.
- UnityEngine.SceneManagement, um die Levels während der Laufzeit zu verwalten.

Ereignisse

Datenmodell

JSON Objekt
Benutzer registrieren

```
{
  "E-Mail": "muster@muster.de",
  "Username": "muster",
  "Password": "pw123"
}
```

JSON Objekt
Benutzer anmelden

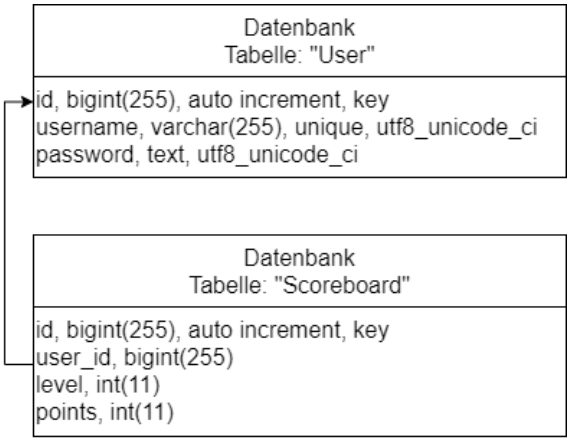
```
{
  "Username": "muster",
  "Password": "pw123"
}
```

JSON Objekt
Punktestand speichern

```
{
  "Username": "muster",
  "Points": "12345"
}
```

JSON Objekt
Punktestand abrufen

```
{
  "Player": [
    "Rank": "1",
    "Username": "muster",
    "Level": "3",
    "Points": "12345"
  ],
  ...
}
```



Abläufe

Entwurf

Fehlerbehandlung

- NullReferenceException
- OutOfMemoryException

Validierung

Projektorganisation

Annahmen
Technologien:
C#, Unity

Git:
Repositories nach Meilensteinen

Entwicklungsumgebung:
Unity (2022.3.26f1), Visual Studio

Erweiterungen:
Ingame Shop mit Skins und DLCs (herunterladbare Inhalte)

Verantwortlichkeiten
Pro Meilenstein
Pro Meilenstein-Unteraufgabe je zwei Personen, wenn möglich (“Pair programming”)

Mögliche Rollen:

Product-Owner: (nicht notwendig für unser Projekt)
Scrum-Master:
Softwarearchitekt: (nicht notwendig für unser Projekt)
Frontend-Entwickler:
Backend-Entwickler:
DevOps-Engineer:

Grober Projektplan
KW 17:

- Projektidee Entwurf
- Grobe Sammlung von Features der Anwendung

KW 18:

- Erstellung Softwarespezifikation
- Team strukturieren

KW 19:

- Einarbeitung in Unity
- Erste Grundkarte (ohne feste Objekte)
- Zwei Bewegliche Objekte (Spieler)

KW 20:

- Hindernisse (Bäume, Wände…)
- NPCs mit movement (ohne einflüsse auf Spieler)
- Texturen/Objekte zeichnen/suchen

KW 21:

- Wetterereignisse

KW 22:

- Erstes Rätsel/Puzzle
- NPCs mit störfaktor

KW 23-25:

- Tests
- Weitere Level
- Spielende

KW 26:

- Datenbank (MySQL)
- Punktestand ermitteln/anzeigen
- Registrier/Login Interface

KW 27-28:

- Feintuning
- Wenn Zeitlich möglich portierung auf Mobile
- Abschluss des Projekts

Anhänge

Glossar

DLC = Downloadable Content / Herunterladbare Inhalte
NPC = NIcht Spieler Charakter

Referenzen

Index