

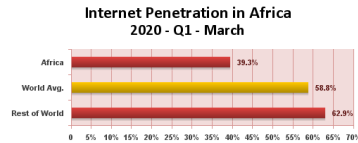
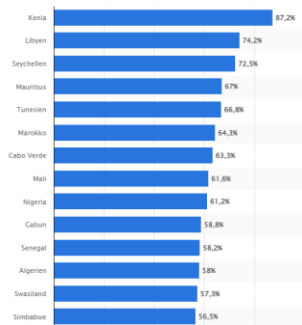
Entwicklungsprojekt WS 21/22

Audit 3



Nachtrag zu Audit 2

- Ausbau des Internets in Afrika unter Durchschnitt
- Einige Länder aber auch deutlich über Durchschnitt
- Wachstum enorm



- Zu POC Absturz: Testen auf Raspberry Pi 4
- Einplatinencomputer mit der Leistung vergleichbar zu älteren Notebooks
- 2 – 8GB RAM
- 4-Kerne-Prozessor mit 1,5GHz
- Ab 50€ (2GB)

Als Nachtrag zum Nutzungskontext unseres Projekts haben wir uns Statistiken über den Ausbau des Internets in Afrika rausgesucht.

Das Internet in Afrika ist zwar noch unterdurchschnittlich ausgebaut, allerdings ist der Wachstum teilweise sehr stark und einige Länder liegen mit dem Ausbau des Internets auch bereits über dem Durchschnitt.

Bei fast 1,4 Milliarden Einwohnern existieren auch bei knapp 40% noch genug mögliche Nutzer für unser Projekt.

Zu unserem POC „Absturz“, also falls durch die schlechte technische Ausrüstung in Afrika unser Programm abstürzt, haben wir uns überlegt, dass man unsere Software auf dem Raspberry Pi 4 testen könnte.

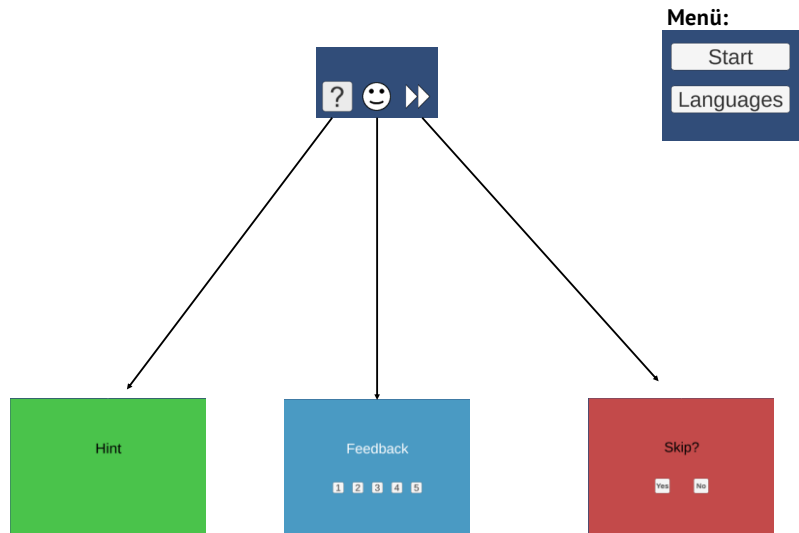
Dieser kleine Computer bietet für nicht so viel Geld eine Leistung, die im besten Falle ausreichen sollte, um unser Lernspiel ausführen zu können. Die günstigste Version ist ab 50€ zu erhalten.



Die Felder werden in zufälliger Reihenfolge gespawnt, aber so dass immer ein richtiges mit einem falschen zusammen ist.



Durchgeführte POCs



Durchgeführte POC's:

Wie im Vorherigen Audit vorgestellt, sind hier unsere Lösungen/Alternativen zu auftretenden Probleme.

Von Links nach Rechts:

- 1.) Spieler kommt nicht weiter, Schwierigkeitsgrad zu hoch
→ Ein Tipp wird angezeigt
- 2.) Spieler kann Level bewerten und eine Erklärung dazuschreiben
→ Um Spiel verbessern zu können und um auf Wünsche einzugehen
- 3.) Spieler kommt nicht weiter, trotz Hinweises, Schwierigkeitsgrad zu hoch
→ Level wird übersprungen

Menü:

- 4.) Nutzer versteht Sprache nicht
→ Nutzer kann im Menü Sprache wählen

Durchgeführte POCs

- Code für POCs „Level-/Spielabschluss“ (Hinweise) und für „Unter-/Überforderung“ (Bewertung)

```
// Hier wird die Methode OnButtonClick definiert
public class OnButtonClick : MonoBehaviour
{
    // Initialisierung der öffentlichen Variablen
    public Image buttonImage;

    // Initialisierung der privaten Variablen
    Button button;
    GameManager gameManager;

    // Start is called before the first frame update
    @UnityMainThread()
    void Start()
    {
        // button und gameManager bekommen die jeweiligen Komponenten zugeordnet
        button = GetComponent<Button>();
        gameManager = GameObject.Find("Game Manager").GetComponent<GameManager>();

        // Wenn wir das Game auf den Button geklickt wird, wird ShowImage() ausgeführt
        button.onClick.AddListener(ShowImage);
    }

    // Update is called once per frame
    @UnityMainThread()
    void Update()
    {
        // Wenn buttonImage inaktiv ist, wird es aktiv oder andersrum
        ShowImage();
    }

    void ShowImage()
    {
        Debug.Log(button.gameObject.name + " was clicked");
        if (buttonImage.gameObject.activeInHierarchy)
            buttonImage.gameObject.SetActive(false);
        else buttonImage.gameObject.SetActive(true);
    }
}
```

- Code für POC „Level-/Spielabschluss“ (Überspringen)

```
public class NextGame : MonoBehaviour
{
    // Initialisierung der privaten Variablen
    Button startButton;

    // Start is called before the first frame update
    @UnityMainThread()
    void Start()
    {
        // Button zuweisen
        startButton = GetComponent<Button>();

        // Bei Drücken des Buttons wird StartGameOnButton ausgeführt
        startButton.onClick.AddListener(StartNextGame);
    }

    // Update is called once per frame
    @UnityMainThread()
    void Update()
    {
        // Methode um Szene des nächsten Spiels zu laden
        StartNextGame();
    }

    void StartNextGame()
    {
        SceneManager.LoadSceneAsync(SceneManager.GetActiveScene().buildIndex + 1);
    }
}
```

Der Code für den Button zu den Hinweisen und dem Button zu der Bewertung des Levels ist der gleiche. Dem zugewiesenen Button wird ein Listener geaddet, der auf einen Klick reagiert. Wenn auf den Button gedrückt wird und das Objekt, das mit der Variable „buttonImage“ definiert wurde, noch nicht aktiviert bzw. noch nicht sichtbar ist, wird dieses aktiviert. Andersrum kann man durch erneutes Drücken des Buttons dieses Objekt auch wieder unsichtbar machen und so das Popup-Fenster wieder schließen.

Für den Skip-Button ist der rechte Screenshot aus dem Code zuständig. Auch hier haben wir wieder mit einem Listener gearbeitet. Wenn auf den Button gedrückt wird, wird die nächste Szene geladen und somit die aktuelle geschlossen bzw. übersprungen.

Durchgeführte POCs

➤ Code für POCs „Absturz“ (Zwischenspeichern)

```
public class GameProgress : MonoBehaviour
{
    // Initialisierung der privaten Variablen
    string progressKey;
    int progress;

    // Start is called before the first frame update
    @UnityMainThread @OnAwake
    void Start()
    {
        // progressKey und progress definieren PlayerPrefs um Spielstand zu speichern
        // bei Start des Spiels wird durch den in den PlayerPrefs gespeicherten Int-Wert die Szene geladen, die zuletzt geöffnet war
        progressKey = "progress";
        progress = PlayerPrefs.GetInt(progressKey, 0);
        if (progress > 0)
        {
            SceneManager.LoadSceneAsync(progress);
        }
    }


    // Update is called once per frame
    @UnityMainThread @OnUpdate
    void Update()
    {
    }
}
```

Für das Zwischenspeichern haben wir die in Unity verfügbaren PlayerPrefs benutzt. Dies ist nur für den Prototyp, da wir im Falle eines echten Projekts durch die großen Datenmengen doch auf eine Datenbank zurückgreifen würden. In einer separaten Klasse wurde der „progressKey“ definiert und es wird die Szene geladen, die in dem PlayerPrefs gespeichert wurde. Beim Laden des ersten Spiels wird der Integer-Wert des PlayerPrefs „Progress“ auf 1 gesetzt, beim Laden des zweiten Spiels auf 2 usw.

Die Klasse „GameProgress“ wird in der Szene „Main Menu“ einem unsichtbaren Objekt (dem Menu Manager) zugewiesen, damit bei Start des Spiels auch immer die „Start“-Methode der Klasse „Game Progress“ aufgerufen wird und somit immer die zuletzt benutzte Szene geöffnet wird.

Anwendungslogik

- Datenbank um Speicherfortschritt der Nutzer zu speichern
- Logindaten speichern um Speicherstände voneinander zu trennen
- Nutzer können nur auf eigenen Account zugreifen
- Lehrperson kann Fortschritte der zugewiesenen Lernenden einsehen
- Webanwendung, da niedrigere Systemanforderungen

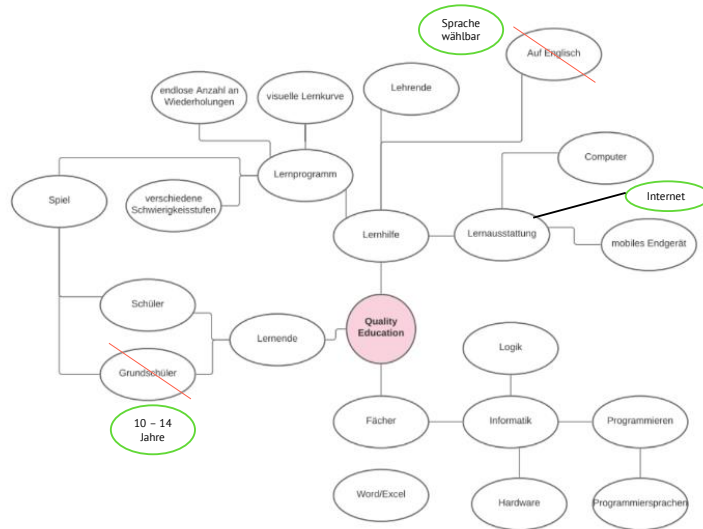


Für die Anwendungslogik haben wir uns dafür entschieden, Logindaten der Nutzer zu speichern. Das muss nur ein Passwort und ein eindeutiger Nutzernamen sein. Eventuell kann man auch seine Schuleinrichtung eintragen lassen. Diese Daten sollten zusammen mit dem dazugehörigen Speicherstand in einer von uns gestellten Datenbank abgespeichert werden. Durch die Logindaten können diese Speicherstände einfacher voneinander getrennt werden.

Ein Nutzer sollte nur auf seinen Account zugreifen können. Eine Lehrperson einer Einrichtung dagegen sollte zusätzlich noch die Möglichkeit haben, die Fortschritte ihrer Schüler und eventuell einige Statistiken über deren Lernverhalten einsehen zu können.

Außerdem haben wir uns letztendlich doch dazu entschieden eine Webanwendung zu erstellen, da das Programm somit auch mit niedrigeren Systemanforderungen benutzt werden kann. Die Logindaten sind somit auch notwendig um die Spielstände zu speichern, da es kein installiertes Programm sein wird, welches einfach Daten auf der Festplatte speichern kann.

Iterierte Modellierungen und Modellierungsbegründungen



In dem Domänenmodell haben wir kleine Iterationen vorgenommen, damit es weiterhin auf dem aktuellen Stand ist. Da wir uns für eine Webanwendung entschieden haben, ist nun Internet eine Voraussetzung der Lernausrüstung und die Sprache soll im Endprodukt wählbar sein, anstatt dass man die Anwendung nur auf Englisch nutzen kann.

Deliverables für den 4. Audit

- Design – Icons selber erstellen oder Presets verwenden
- Auf Anzahl und Schwierigkeitsgrad der Level einigen
- Auf Art des Speicherns der Spielstände einigen (Eher Serialisierung + JSON)
- Checken ob alle gesetzten Ziele erreicht wurden und notfalls iterieren
=> Finaler Prototyp



Quellen

- [Africa Internet Users, 2021 Population and Facebook Statistics \(internetworldstats.com\)](https://internetworldstats.com)
- [Afrika - Anteil der Internetnutzer 2020 | Statista](https://www.statista.com/de/statistik/datenbanken/afrika.html)
- [Buy a Raspberry Pi 4 Model B – Raspberry Pi](https://www.buyzero.de/raspberry-pi-4-model-b)
- [Raspberry Pi 4B 2GB/4GB \(buyzero.de\)](https://www.buyzero.de/raspberry-pi-4-model-b)
- [Raspberry Pi 4 Modell B \(2 GB RAM\), 49,90 € - Welectron](https://www.welectronic.com/raspberry-pi-4-model-b)
- <https://www.raywenderlich.com/418-how-to-save-and-load-a-game-in-unity#toc-anchor-002>

