Praktikum GUI-Development – Aufgabe 2

Prof. Dr. Matecki

Ausgabe des Aufgabenblatts: 11.12.2023
Abgabe per ILIAS bis Sonntag, 21.01.2024 10:00 Uhr
Abnahme: an dem Praktikumstermin nach dem Abgabetermin

- Zulässig zur Bearbeitung dieses Aufgabenblatts sind Teams mit 1-3 Studierenden.

- Entwürfe müssen als wissenschaftliche Ausarbeitung formuliert sein
- Eine wissenschaftliche Ausarbeitung enthält in diesem Praktikum immer

    - Ein Deckblatt mit Fach und den Namen der Team-Mitglieder

    - Ein Inhaltsverzeichnis vor Beginn der Ausarbeitung

    - Ein Kapitel für jede Aufgabe mit sinnvollen Kapitelüberschriften
        - Ein Unterkapitel für jede Unteraufgabe
        - ... mit sinnvollen Kapitelüberschriften !!

    - Auf der Fußzeile die Seitenzahl

- Tipp: Die in der Vorlesung besprochenen Programmbeispiele sind im Rahmen Ihrer Lösungen zur Nutzung erlaubt!

- Angeschaut werden von den Lehrenden

    - Ihre Dokumentation mit:
        - UML-Klassendiagramm für die SW-Architektur
        - Workflows als UML-Aktivitätsdiagramm
        - Screenshot mit Ablaufnachweis
    - Ihre SW / Ihr Quellcode als AndroidStudio-Projekt
    -

- Die Implementierung erfolgt unter Android Studio
- API-Level 33 sollte mindestens im Emulator getestet worden sein.

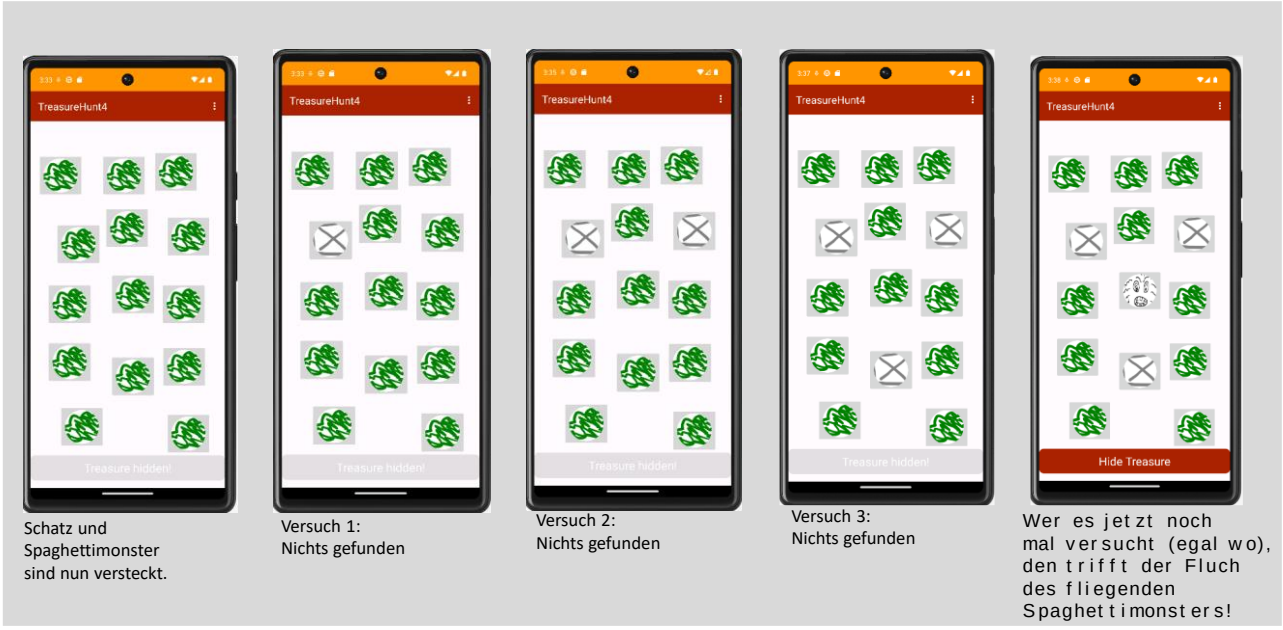
Entwicklung einer Android-App für eine „Schatzsuche“

Ihre App enthält beim Start eine **AppCompatActivity** mit folgender Spiel-Ansicht in einem ConstraintLayout:

Das Spiel funktioniert im Play-Fragment / Play-Bereich folgendermaßen:

Die ImageButtons des „Waldes“ sind initial insensitiv.

Sie beginnen das Spiel, indem Sie den Button „Hide Treasure“ drücken. Er wird danach insensitiv geschaltet. Die ImageButtons des „Waldes“ werden dagegen sensitiv geschaltet:



Schatz und Spaghettimonster sind nun versteckt.

Versuch 1: Nichts gefunden

Versuch 2: Nichts gefunden

Versuch 3: Nichts gefunden

Wer es jetzt noch mal versucht (egal wo), den trifft der Fluch des fliegenden Spaghettimonsters!

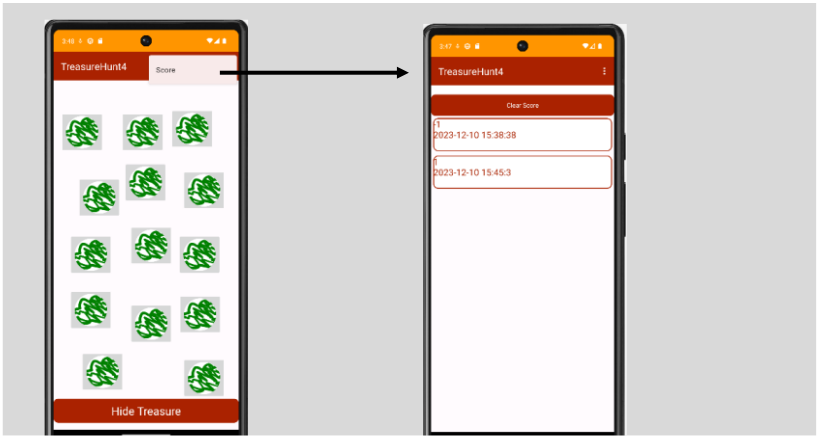
Der Fluch des fliegenden Spaghettimonsters kann Sie auch während der 3 Versuche treffen, sofern Sie das Feld erwischen, hinter dem es sich versteckt.

Wenn der Schatz gefunden wird, so sieht das folgendermaßen aus:

Schatz und Spaghettimonster sind nun versteckt.

Versuch 1: Schatz gefunden!

Mit dem Menü rechts oben können Sie eine SubActivity starten, die **maximal Ihre letzten 5 Scores** anzeigt:



- Jeder Score-Eintrag enthält
  - den Zeitstempel (Datum/Uhrzeit) der letzten Spiele, sowie die Anzahl der benötigten Spielzüge zum Gewinnen in einer Liste .
  - Bei Verlust: statt der Anzahl der Spielzüge einfach „-1“.

- Die „SpielButtons“ sind kleine ImageButtons mit einer Grafik, die „Wald“ symbolisiert.
- Daneben gibt es noch Grafiken, die „Schatz“, „nichts gefunden“ oder „Monster“ symbolisieren.
- Andere Analogien mit **eigenen Symbolen** („Meer“,“Schatzinsel“,“…“) sind erlaubt und gewünscht!
- Die Grafiken gestalten Sie selbst, ebenso die genauen Positionen der Spielfelder.

- **Die Spiellogik ist außerhalb der Activity in einer oder mehreren eigenen Klasse/n unterzubringen!**
- **Der Start der SubActivity erfolgt nach der neuen Technik mit einem Launcher, wie in der Vorlesung gezeigt!**

Ausgebaute Variante für Teams mit 2-3 Personen:

- Die Spiellogik ist in einem ViewModel unterzubringen
- Das Spielfeld ist in einem Fragment unterzubringen, welches von der onCreate()-Methode eingeblendet wird
- Verwenden Sie für die einzelnen Werte innerhalb des Spiels eigene **MutableLiveData**-Variablen, die Sie vom Darstellungsfragment aus per Observer „beobachten“!

Bitte beachten Sie einige technische Details:

- Für die unterschiedlichen Symbole sind eigene Grafiken zu entwickeln.
- Sämtliche Titel-Strings sind in die Datei *strings.xml* auszulagern
- Die Listenansicht der Scores können Sie (freie Wahl)
    - entweder über die (ältere) *ListView*
    - oder eine *RecyclerView* realisieren. Bei der *RecyclerView* müssen Sie einen eigenen *Adapter* für die Datenhaltung implementieren.
    - Eine rudimentäre Datenzugriffs-Klasse wird Ihnen zur Verfügung gestellt. Diese kann (und soll) für Ihre Zwecke überarbeitet werden.
    - Die *TextView*-Einträge der Liste müssen als Hintergrund **eine selbst definierte *Shape*** enthalten.
    - Hinterlegen Sie ein eigenes Theme mit eigenen Farbkompositionen.
- Der Datei-Zugriff für die Scores erfolgt **App-intern** – ist also von außen nicht sichtbar. (SAF oder MediaStore wird also nicht benötigt).
- Sie dürfen das *ConstraintLayout* des Spiels gern auch mit komplexeren Wegen gestalten! Die Variante im Aufgabenblatt ist nur als Beispiel gedacht

Es ist eine Dokumentation anzufertigen die mindestens enthält:

- Titelblatt mit Name/n des/der Autoren/Autorinnen.
- Inhaltsverzeichnis
- Die Darstellung Ihrer einzelnen Ansichten des Spiels als Screenshots
- Die Darstellung der wesentlichen Abläufe als UML2-**Aktivitätsdiagramm**
- Die Darstellung der wesentlichen Klassenbeziehungen als UML2-Klassendiagramm.