Designentscheidungen für das Minigame *Muster-Spiel*

Das Minigame Muster-Spiel ist in das Paket **„muster“ ausgelagert** und exportiert nur die Hauptfunktion, die den Spielaufbau und -ablauf steuert. Alle weiteren Funktionen werden so vor dem Hauptprogramm verborgen, sodass keine Kollisionen entstehen. Die Rückgabe erfolgt als *punkte* (uint32) und *note* (float32) zurück an das Hauptspiel.

Innerhalb des Spieles existieren **neben der Hauptroutine** weitere **vier nebenläufige Routinen**, die im Folgenden näher erläutert werden. Dabei übernimmt die Hauptroutine selbst die Rolle der **Tastaturabfrage** nachdem die übrigen Routinen gestartet sind. Damit gewährleistet ist, dass die nebenläufigen Routinen nach Beenden der Hauptroutine nicht im Hauptprogramm weiterlaufen, wurde eine Wait-Group eingerichtet. Die Kommunikation zwischen den Routinen geschieht über geteilte Adressen von, vor allem Bool-Werten und einen gemeinsamen Channel.

Muster-Spiel nutzt die sehr umfangreiche Klasse **„objekte“** für klickbare, angezeigte Elemente und lagert komplexere Texte und Text-Arrays/Slices in der Klasse **„texte“** aus. Für den gegenseitigen Ausschluss, vor allem beim Zeichnen aller Objekte, werden Mutexe aus der Klasse **„sync“** genutzt.

Verwendete Routinen:

- **Tastaturabfrage:** Go-Hauptroutine als for-Schleife, die über das ganze Spiel hinweg auf Eingaben wartend läuft, jedoch nur in den Eingabe-Teilen der Spiele freigeschaltet ist (über den bool *tastatur*).

- **view_komponente:** Die nebenläufige Go-Routine, die alleine für die visuelle Darstellung zuständig ist. Sie zeichnet, also aktualisiert ständig die Maus, die als Bild angezeigt wird. Alle anderen Elemente, die auf den Hintergrund gezeichnet sind werden aus Performance-Gründen einmalig archiviert und ständig restauriert. Nur wenn neue Objekte erscheinen oder verschwinden wird der Hintergrund einmalig im Hintergrund (gfx-Update) gezeichnet und wieder archiviert. Die for-Schleife wird durch adaptive Verzögerung auf etwa 100 FPS gehalten.

- **spielablauf:** Die nebenläufige Go-Routine, die alle Spielelteile und -funktionen zeitlich plant und nacheinander einsetzen lässt. Über Bools und den Channel erhält die Routine die Signale zum fortschreiten in folgende Abschnitte von hauptsächlich der Maussteuerung. Im spielablauf werden alle Objekte erstellt und damit im geteilten Objekt-Slice (*obj) für View-Komponente und Maussteuerung verfügbar gemacht, sodass Anzeige und Interaktion ermöglicht werden.

- **maussteuerung:** Die nebenläufige Go-Routine, welche die Maus-Koordinaten ständig aktualisiert (und an restliche Routinen weitergibt) und nach Maus-Klicks auf Treffer von, an diese Routine übergebenen, Objekten überprüft und entsprechend Signale an restliche Routinen sendet bzw. Sounds abspielt.

- **musikhintergrund:** Die nebenläufige Go-Routine, welche für die Hintergrund-Musik einen Loop eines Musikausschnitts abspielt, bis die anderen Routinen beendet werden.

Verwendete Klassen:

- Die **Klasse „objekte“** wurde so geschrieben, dass sie diverse Funktionen vereint. Sie ist sehr umfangreich und flexibel für andere Spiele nutzbar. Beinhaltet sind die Disigns aller dargestellten Objekte, die als Bilder geladen oder über gfx gezeichnet werden. Über die Methoden der Klassen sollten sämtliche Operationen der Objekte gesteuert werden können. Dies beinhaltet ihre Lage (Koordinaten) im gfx-Fenster, die Aktivität, ihre dargestellte Größe, ihren Typ und ihre Erstellungszeit, einen optionalen Text (string) zur Beschriftung und eine Bedingung für eingegebene Koordinaten, sodass das entsprechende Objekt „getroffen“ worden ist.

- Die Klasse **„texte“** enthält als zu exportierende Variablen Strings und Arrays/Slices von Strings, die im Hauptprogramm die Übersichtlichkeit verbessern sollen.

- Weitere Klassen (für z. B. Zwischenanzeigen, Berechnung der Endnote oder die Anzege des Enbildschirms) hätten separat erzeugt werden können. Da aber von diesen **Zusatzfunktionen** auf diverse geteilte Variablen zugegriffen wird und diverse Interaktionen innerhalb stattfinden sollen, wurde von einer Auslagerung abgesehen.

UML-Diagramm zum muster-Paket: