Human-Computer Interaction Group

Leibniz Universität Hannover

Übungsblatt 2

Interaktive Systeme – SoSe 24

Prof. Dr. Michael Rohs, Jan Feuchter, M.Sc.

Alle Übungen, die nicht explizit als Gruppenaufgabe deklariert sind, müssen in Einzelarbeit geleistet und individuell abgegeben werden. Identische Abgaben werden als Plagiat behandelt.

Abgabe bis Montag den 15.04. um 23:59 Uhr über https://assignments.hci.uni-hannover.de/SoSe2024/ISy. Die Abgabe muss aus einer einzelnen zip-Datei bestehen, die alle nötigen Dateien enthält. Lösen Sie Umlaute in Dateinamen bitte auf.

Verpacken Sie Ihre Lösung (PDF-Datei zu Aufgaben 1, 2 und 3, sowie Ihr exportiertes Projekt als zip-Datei zu Aufgabe 4) zum Hochladen in eine zip-Datei. Nutzen Sie dabei z.B. die Export Funktionalität von IntelliJ, um zu gewährleisten dass alle benötigten Dateien in ihrer Abgabe vorhanden sind.

Aufgabe 1: Maus mit Scrollrad (20 Punkte)

Viele Computermäuse besitzen ein sogenanntes Scrollrad (siehe Abbildung). Das Scrollrad kann vorwärts und rückwärts rotiert und (wie eine Taste) heruntergedrückt werden.

a) Geben Sie die Zustände und Aktionen für das Scrollrad an. Ignorieren Sie Zustände in denen simultane Aktionen möglich sind (bspw. Drücken des Rades und gleichzeitiges Scrollen nach unten).

b) Zeichnen Sie das Zustandsübergangsdiagramm für das Scrollrad.

(Jeweils nur für das Scrollrad, nicht für die komplette Maus.)

Aufgabe 2: Instrumental Interaction (18 Punkte)

In der Vorlesung wurde das Konzept der „Instrumental Interaction" vorgestellt. Das zugehörige Paper findet sich Online[1]. Beschreiben Sie kurz für die folgenden GUI-Elemente die Eigenschaften „degree of indirection", „degree of integration" und „degree of compatibility":

a) Menü

b) Dialogbox

c) „Handles" für das grafische Editieren von Objekten in Präsentations- oder Zeichenprogrammen

Aufgabe 3: GUI Design für ältere Menschen (11 Punkte)

Mit den besprochenen Layout Panes können Buttons automatisch gelayouted werden. Die Abstände zwischen zwei Buttons können dabei frei variiert werden. Sie haben nun die Aufgabe die Größe der Buttons und deren Abstand für ein grafisches Interface auf einem Touchscreen anzupassen. Die Nutzergruppe soll aus älteren Menschen bestehen. Lesen Sie hierfür das verlinkte Paper[2] und beantworten Sie folgende Fragen:

a) Welche Größe eines einzelnen Buttons würden Sie wählen, wenn eine schnelle Reaktionsgeschwindigkeit erreicht werden soll? Welche Größe von mehreren nebeneinander angeordneten Buttons würden Sie wählen, wenn die Displaygröße begrenzt ist?

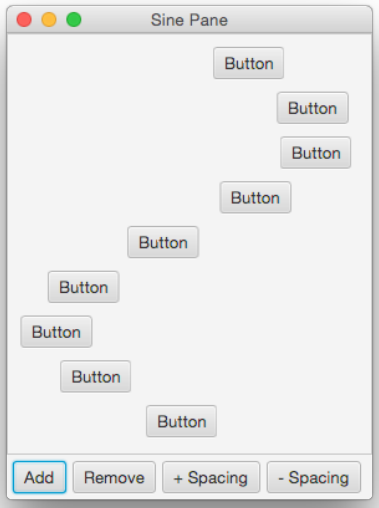
b) Welchen Abstand (Spacing) zwischen den Buttons würden Sie wählen, wenn mehrere Buttons nebeneinander angeordnet werden sollen? Was passiert, wenn der Abstand zu groß gewählt wird?

c) Nennen Sie mindestens zwei Beispiele für grafische User Interfaces, die alle Altersgruppen mit Beginn der Volljährigkeit bis hin zu älteren Menschen nutzen. Welche Altersgruppe ist hauptsächlich beim Design zu beachten?

[1] Michel Beaudouin-Lafon. Instrumental Interaction: An Interaction Model for Designing Post-WIMP User Interfaces. Proceedings of CHI 2000, pp. 446-453.
https://www.lri.fr/~mbl/papers/CHI2000/

[2] Xia Jin, Zhao & Plocher, Thomas & Kiff, Liana. (2007). Touch Screen User Interfaces for Older Adults: Button Size and Spacing. Conference: Universal Access in Human Computer Interaction. Coping with Diversity, 4th International Conference on Universal Access in Human-Computer Interaction, UAHCI 2007. 933-941.
https://link.springer.com/chapter/10.1007/978-3-540-73279-2_104

Aufgabe 4: Layout Panes (optional, 20 Bonuspunkte)



Zur Lösung dieser Aufgabe (und einiger zukünftiger Aufgaben) wird das OpenJDK, JavaFX, sowie IntelliJ IDEA als Entwicklungsumgebung benötigt. Installieren Sie diese Komponenten auf Ihrem Rechner. In der Regel können Sie über die IDE auch die anderen benötigten Komponenten beziehen.

Auf Stud.IP finden Sie ein Template (`SineLayout.zip`) für IntelliJ IDEA. Im Quelltext sind die zu bearbeitenden Stellen mit „`@todo`" markiert. Schreiben Sie Ihre Antworten zu den Teilaufgaben als Kommentare in den Quelltext.

Die Klasse `SinePane` zum Layout von Knoten (Nodes) ist von `javafx.scene.layout.Pane` abgeleitet und soll folgende Layout-Strategie implementieren:

Die Knoten werden auf ihre präferierte Größe skaliert und von oben nach unten angeordnet. Die x-Positionen werden so gewählt, dass die Mitten der Knoten auf einer Sinuskurve liegen, die vertikal verläuft. Zwischen den Knoten kann ein wählbarer Freiraum (spacing) eingestellt werden. An den Rändern der `SinePane` befindet sich ebenfalls ein Freiraum (padding). Die Knoten sollen nicht unter ihre präferierte Größe herunterskaliert werden.

a) Erklären Sie kurz, was geschieht, wenn der Add-Button angeklickt wird.

b) Erklären Sie kurz, warum die Methode `setSpacing` die Methode `requestLayout` aufruft. Was geschieht, wenn `requestLayout` hier nicht verwendet wird?

c) Erläutern Sie die Methode `double waveX(double y)`. Wie wird die x-Position aus der als Parameter übergebenen y-Position berechnet?

d) Beschreiben Sie die Funktionsweisen der Methoden `computePrefHeight` und `computePrefWidth`.

e) Beschreiben Sie im Kommentar der Methode `computeMaxWidth` was passieren würde wenn max width und preferred width den selben Wert haben.

f) Vervollständigen Sie die Dokumentationen der Funktionen `maxPrefWidthChildren`, `sumPrefHeightChildren`, `sumSpacing`.

g) Vervollständigen Sie die Implementierung von `layoutChildren`. Die Knoten werden mit `autosize` zunächst auf ihre präferierten Größen skaliert und sollen nun positioniert werden wie in der obigen Abbildung gezeigt. (Etwaige Layout-Attribute der Knoten sollen ignoriert werden.)

Die Dokumentation der Klasse Pane (Basisklasse für Layout-Panes) finden Sie unter:
https://openjfx.io/javadoc/11/

Exportieren Sie Ihre Lösung als zip-Datei aus IntelliJ und geben Sie diese mit ab.