TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Návrh prvků uživatelského rozhraní ve vývojovém prostředí Processing

Bakalářská práce

Studijní program: B2646 – Informační technologie
Studijní obor: 1802R007 – Informační technologie

Autor práce: **Lukáš Hradecký**
Vedoucí práce: Ing. Tomáš Martinec, Ph.D.

Liberec 2017

Technická univerzita v Liberci
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: **2016/2017**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Lukáš Hradecký**
Osobní číslo:	**M14000029**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Návrh prvků uživatelského rozhraní ve vývojovém prostředí Processing**
Zadávající katedra:	**Ústav mechatroniky a technické informatiky**

Zásady pro vypracování:

1. Seznamte se se strukturou a funkcemi standardních grafických prvků uživatelského rozhraní v OS Windows a Android.
2. Navrhněte vlastní strukturu prvků uživatelského rozhraní, kterou by bylo možné realizovat v prostředí Processing a které by bylo možné používat pro OS Windows i Android.
3. Realizujte takovou množinu základních i pokročilejších prvků uživatelského rozhraní z bodu 2, která umožní snadno vytvářet uživatelsky příjemné přenositelné programy.
4. Vytvořte pro realizované prvky dokumentaci a sadu příkladů.

Rozsah grafických prací: **dle potřeby dokumentace**

Rozsah pracovní zprávy: **30–40 stran**

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

[1] **Casey Reas, Ben Fry: Make: Getting Started with Processing, September 2015, Maker Media. 238 pages.**

[2] **Casey Reas, Ben Fry: Processing: A Programming Handbook for Visual Designers, December 2014, The MIT Press. 720 pages.**

[3] **Lukáš Marek: Programování pro Android v příkladech, seriál, [online] http://www.root.cz/serialy/programovani-pro-android-v-prikladech/#ic=serial-box&icc=more**

Vedoucí bakalářské práce: **Ing. Tomáš Martinec, Ph.D.**
Ústav mechatroniky a technické informatiky

Konzultant bakalářské práce: **Ing. Přemysl Svoboda**
Ústav mechatroniky a technické informatiky

Datum zadání bakalářské práce: **10. října 2016**

Termín odevzdání bakalářské práce: **15. května 2017**

Technická univerzita v Liberci - 14 -

L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

doc. Ing. Milan Kolář, CSc.
vedoucí ústavu

V Liberci dne 10. října 2016

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum: 15. 5. 2017

Podpis: Hradecký

Poděkování

Tímto bych chtěl poděkovat svému vedoucímu panu Ing. Tomáši Martincovi, Ph.D. za odborné vedení a poskytnuté rady při zpracování práce.

Abstrakt

Tato práce se zabývá návrhem struktury grafických ovládacích prvků uživatelského rozhraní, které by bylo možné realizovat ve vývojovém prostředí Processing a používat je v operačních systémech Windows a Android. V práci jsou uvedeny jednotlivé komponenty v těchto systémech, ze kterých je vybrána množina nejpoužívanějších prvků, jenž lze používat v multiplatformních aplikacích. Dále práce obsahuje návrh grafické podoby těchto vybraných prvků a také popis každého z nich.

Následně se práce věnuje realizaci některých z těchto navržených prvků. Tato realizovaná množina umožňuje snadno vytvářet přenositelné aplikace použitelné ve zmíněných operačních systémech. Praktická část práce také zahrnuje popis důležitých částí tvorby prvků, mezi něž patří reakce na vstup od uživatele a princip líného vykreslování. Dále práce popisuje realizaci jednotlivých prvků a také jejich vzájemnou strukturu.

Součástí práce je také vytvoření knihovny s těmito realizovanými prvky. Tato knihovna obsahuje sadu příkladů, jejichž popis je v práci uveden spolu s jejich grafickou podobou a ukázkou kódu v prostředí Processing. V textu je také uveden popis tohoto vývojového prostředí a jeho funkcí. Jelikož výstupem práce je zmíněná knihovna s realizovanými prvky, práce obsahuje také popis již existujících knihoven s prvky uživatelského rozhraní.

Klíčová slova

uživatelské rozhraní, vývojové prostředí Processing, grafické ovládací prvky, Windows, Android

Abstract

This text describes design of structure of the graphic user interface controls that could be implemented in development environment Processing and could be used in operating systems Windows and Android. In the thesis are listed individual components in these systems, from which a set of the most used elements, which can be used in multiplatform applications, is selected. Further, the thesis contains a graphic design of these selected elements and a description of each of them.

Subsequently, the thesis deals with realization of some of these proposed elements. This implemented set allows easy creation of portable applications usable in said operating systems. The practical part of the thesis also includes a description of important parts of element creation, including reaction to user input and the principle of lean rendering. Further, the thesis describes the realization of individual elements as well as their mutual structure.

Part of the thesis is also the creation of a library with these realized elements. This library contains a set of examples, the description of which is presented in the work together with their graphic form and sample code in the environment Processing. The text also describes this development environment and its functions. As the output of the work is the mentioned library with implemented elements, the work also contains a description of already existing libraries with elements of the user interface.

Keywords

user interface, development environment Processing, graphical controls, Windows, Android

Obsah

Seznam obrázků

Seznam zdrojových kódů

Úvod

Tuto práci jsem si vybral, protože mě zaujal vývoj v prostředí Processing. S tímto vývojovým nástrojem jsem se dříve nesetkal a chtěl jsem vyzkoušet, jak se s ním pracuje. Proto jsem se s tímto prostředím seznámil a prozkoumal jeho funkce. Po vyzkoušení dostupných knihoven s prvky uživatelského rozhraní jsem zjistil, že některé z nich nejsou kompatibilní se současnou verzí prostředí Processing nebo je nelze použít k vytváření aplikací pro systém Android. Proto je v této práci mým cílem navrhnout strukturu prvků tak, aby byla použitelná pro systém Windows i Android.

Pro potřeby návrhu jsem zkoumal strukturu a funkce standardních grafických prvků uživatelského rozhraní v těchto operačních systémech. Přemýšlel jsem nad tím, jaké typy těchto prvků jsou nejčastěji používané k jejich ovládání. Pro realizaci jsem vybral některé z mých navržených prvků a implementoval je ve vývojovém prostředí Processing. Takto vytvořená knihovna obsahuje i sadu jednoduchých příkladů použití obsažených prvků.

1 Vývojové prostředí Processing

1.1 Popis

Processing je open source programovací jazyk a integrované vývojové prostředí, které bylo vytvořeno s cílem výuky základů programování ve vizuálním kontextu.[1] Jedním ze stanovených cílů Processingu je působit jako nástroj, prostřednictvím něhož lze uživatele, kteří dříve neprogramovali, přivést k programování prostřednictvím okamžitých viditelných výsledků skrz vizuální zpětnou vazbu. Tento jazyk vychází z programovacího jazyka Java, ale používá zjednodušenou syntaxi a grafický programovací model.[1]

Hlavní výhodou tohoto prostředí je multiplatformní vývoj, lze v něm vytvářet desktopové aplikace, webové aplikace nebo lze cílit na mobilní platformu. Processing je dostupný pro operační systémy Windows, Max OS X a GNU/Linux.

Tento projekt zahájili v roce 2001 Casey Reals a Benjamin Fry. V roce 2012 založili Processing Foundation spolu s Danielem Shiffmanem, který se formálně připojil jako třetí vedoucí projektu.[1]

1.2 Funkce

Processing obsahuje tzv. sketchbook, což je minimální alternativa integrovaného vývojového prostředí pro organizování projektů. Každý sketch je podtřída třídy PApplet v jazyce Java, která implementuje většinu funkcí jazyka Processing.[1]

Při programování v Processingu je před kompilací kód přeložen do čistého jazyka Java a při překladu se se všemi definovanými třídami zachází jako s vnitřními třídami. To znamená, že je zakázáno použití statických proměnných a metod, pokud není v Processingu nastaveno, že je program psaný v čistému Java módu. Processing také umožňuje uživatelům vytvářet jejich vlastní třídy uvnitř sketche PApplet.[1]

1.3 Příklad kódu

```
void setup() {
        size(400, 400); // Nastavení velikosti okna.
        stroke(255);  //  Nastaví  barvu  obrysu  následujících
objektů.
        background(100, 100, 100); // Nastaví barvu pozadí.
}
void draw() {
        line(100, 50, mouseX, mouseY);
        // Vykreslí čáru z bodu[100,50] do pozice kurzoru.
}
```

Zdrojový kód 1: Příklad zdrojového kódu programovacího jazyka Processing

1.4 Processing pro Android

Projekt, jehož primárním cílem je umožnit velmi jednoduchou tvorbu aplikací pro operační systém Android prostřednictvím Processing API. Pokud chce uživatel vytvářet aplikace pro mobilní platformu, kromě samotného Processingu k tomu potřebuje také Android SDK pro verzi systému Android, na kterou cílí vytvářené aplikace. Pro samotné programování pak už jen stačí zvolit mód Android. Pokud má uživatel již hotovou aplikaci pro Windows v Processingu, nemusí nic přepisovat, kód se automaticky převede při kompilaci.

Tvorba aplikace se liší pouze v nabídkách spuštění a exportu, kde jsou na výběr tyto možnosti:

- spuštění v emulátoru
- spuštění na zařízení se systémem Android připojeném přes USB
- exportování jako Android projekt
- exportování podepsaného balíčku

1.5 Existující knihovny s prvky uživatelského rozhraní

1.5.1 ControlP5

Tuto knihovnu vyvíjí Andreas Schlegel od roku 2006.[3] Obsahuje velké množství prvků (nejvíce ze všech těchto knihoven), od nejpoužívanějších až po méně časté. Většina těchto prvků funguje v Android módu, proto je lze používat i v tomto mobilním systému.

Nevýhodou této knihovny je to, že je vyvíjena pro starší verzi prostředí Processing, a tudíž některé z prvků v nejnovější verzi nepracují správně. I přesto je to vzhledem k délce vývoje nejpoužívanější knihovna prvků uživatelského rozhraní.

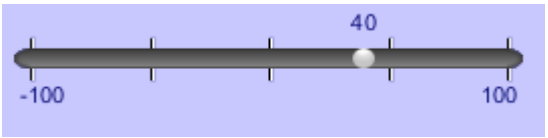
Obrázek 1: Vzhled tlačítka v knihovně ControlP5

```
// příklad definice tlačítka
import controlP5.*; ControlP5 cp5;
cp5 = new ControlP5(this);
cp5.addButton("colorA")
     .setValue(0)
     .setPosition(100,100)
     .setSize(200,19)
     ;
// metoda, která vypíše název prvku, jehož akci zachytila
public void controlEvent(ControlEvent theEvent) {
  println(theEvent.getController().getName());
  n = 0;
}
// metoda, která zachytí konkrétní prvek (zde tlačítko)
public void colorA(int theValue) {
  println("a button event from colorA: "+theValue);
}
```

Zdrojový kód 2: Příklad vytvoření tlačítka v knihovně ControlP5

1.5.2 G4P

Knihovna vyvíjená Peterem Lagerem.[4] Obsahuje o něco méně prvků než knihovna ControlP5, ale vynahrazuje to kompatibilitou s nejnovější verzí Processingu. Autor také vytvořil nástroj, který využívá tuto knihovnu, a pomocí něhož lze vizuálně umísťovat prvky do okna a měnit jejich vlastnosti. Nevýhodou je absence možnosti vytvářet pomocí této knihovny mobilní aplikace pro Android.



Obrázek 2: Vzhled posuvníku v knihovně G4P

```
// příklad definice posuvníku
import g4p_controls.*;
GCustomSlider sdr1;
sdr1 = new GCustomSlider(this, 20, 20, 260, 50, null);
  sdr1.setShowDecor(false, true, true, true);
  sdr1.setNbrTicks(5);
  sdr1.setLimits(40, -100, 100);
// metoda, která vypíše hodnotu prvku, jehož akci zachytila
void handleSliderEvents(GSlider slider) {
  println("integer  value:"  +  slider.getValueI()  +  "  float
value:" + slider.getValueF());
}
```

Zdrojový kód 3: Příklad vytvoření posuvníku v knihovně G4P

1.5.3 Guido

Knihovnu vytvořil Florian Jenett.[5] Je to jednoduchá knihovna, kterou lze použít při vytváření desktopových aplikací. Obsahuje ale výrazně méně prvků než předchozí knihovny. Nevýhoda spočívá v tom, že není kompatibilní s nejnovější verzí.

Obrázek 3: Vzhled tlačítka v knihovně Guido

```
// příklad definice tlačítka
import de.bezier.guido.*;
SimpleButton button;
// vytvoření manažera, který obsluhuje stisk tlačítka
Interactive.make( this );
// vytvoření tlačítka
new SimpleButton( 50, 50, 100, 100);
```

Zdrojový kód 4: Příklad vytvoření tlačítka v knihovně Guido

1.5.4 Interfascia

Vytvořena Brendanem Bergem.[5] Obsahuje podobný počet prvků jako knihovna Guido. Poslední verze pochází z roku 2016, tudíž je kompatibilní se současnou verzí prostředí Processing.

Obrázek 4: Vzhled tlačítka v knihovně Interfascia

```
// příklad definice
import interfascia.*;
GUIController c;
IFButton b1;
// vytvoření tlačítka a nastavení odchycení
c = new GUIController (this);
b1 = new IFButton ("Green", 30, 35, 60, 30);
b1.addActionListener(this);
c.add (b1);
// metoda pro odchycení stisku tlačítka
void actionPerformed (GUIEvent e) {
  if (e.getSource() == b1) {
    background(100, 155, 100);
  }
}
```

Zdrojový kód 5: Příklad vytvoření tlačítka v knihovně Interfascia

2 Standardní grafické prvky uživatelského rozhraní

2.1 Popis

Grafické prvky uživatelského rozhraní jsou prvky, které umožňují interakci programu s uživatelem. Každý prvek napomáhá specifickému typu této interakce a je vizuálně ztvárněn jako viditelná součást grafického rozhraní aplikace.[2] Charakteristickou vlastností ovládacího prvku je poskytování interakce pro přímou manipulaci s daným typem dat. Jinými slovy, komponenty v aplikaci slouží jako stavební bloky, které tuto manipulaci umožňují. Některé prvky podporují interakci s uživatelem, další se chovají jako kontejnery, které v sobě seskupují jiné prvky.[2]

V kontextu aplikace mohou být komponenty v daný okamžik povolené nebo nepovolené. Povolený prvek může reagovat na akce uživatele. Prvek, který na tyto akce nedokáže reagovat, je považován za nepovolený. Povolený prvek má obvykle odlišný vzhled od nepovoleného.[2]

V následujících kapitolách popíši komponenty v operačních systémech Windows a Android. Popis těchto prvků jsem rozdělil na dvě části podle systému, protože uživatelské rozhraní jednoho systému se chová a ovládá jinak než rozhraní druhého. Prvky v systému Windows jsou uzpůsobeny pro klikání kurzorem, kdežto systém Android má prvky přizpůsobeny ovládání dotykem a s ním spojenými gesty. Proto jsem při návrhu struktury vybíral prvky, které se dobře ovládají jak klikáním, tak i dotykem. S tím souvisí grafický návrh prvků, který je přizpůsobený právě ovládání obojího typu.

2.2 Standardní prvky systému Windows

Systém Windows obsahuje velké množství ovládacích prvků uživatelského rozhraní. Některé jsou čistě popisné, jiné umožňují interakci s uživatelem a další například práci s textem. Podrobněji se zde budu věnovat prvkům knihovny Windows Forms v platformě Microsoft .NET Framework. Příloha B[6] ilustruje strukturu těchto prvků. V následujícím seznamu si popíšeme jednotlivé prvky. Začneme viditelnými.

- *Tlačítko* (Button) - po kliknutí vykoná akci
  - Zaškrtávací *tlačítko* (CheckBox) - po kliknutí povolí nebo nepovolí možnost, lze zaškrtnou více ve skupině
  - *Přepínač* (RadioButton) - slouží pro výběr jedné z více možností (v páru nebo ve větší skupině), vybraná je vždy pouze jedna možnost
- *Nabídka* (Menu) - seznam možností nebo příkazů, které se vykonají po kliknutí
  - *Kontextová nabídka* (ContextMenu) - typ nabídky, jejíž obsah závisí na kontextu nebo stavu, kdy je nabídka vyvolána
  - *Lišta nabídek* (MainMenu) - obsahuje více nabídek v liště
  - *Položka nabídky* (MenuItem)
- *Seznam obrázků* (ImageList)
- *Stavový řádek* (StatusBarPanel) - zobrazuje informace, obvykle se nachází na spodním okraji okna
- *Kontextová nápověda* (ToolTip)- informační okno, které se zobrazí po najetí myší na jiný prvek
- *Mřížka* (DataGrid) - data jsou uspořádána do řádků a sloupců
- *Skupinový rámec* (GroupBox) - seskupení více prvků, může mít popisek
- *Popisek* (Label) - text, který popisuje jiný prvek
- *Odkaz* (LinkLabel) - hypertextový odkaz v popisku
- *Výběrové pole* (ComboBox) - textové pole s připojenou nabídkou nebo seznamem, uživatel může přidat novou hodnotu do pole, nebo může vybrat již existující hodnotu
- *Seznam* (Listbox) - umožňuje vybrat jednu či více položek se seznamu

- *Zaškrtávací seznam* (CheckedListBox) - každá položka seznamu obsahuje zaškrtávací tlačítko
- *Seznam* (ListView) - více možností zobrazení položek než klasický seznam
- *Položka seznamu* (ListViewItem) - představuje položku v seznamu
- *Obrázek* (PictureBox) - slouží pro zobrazení obrázků
- *Indikátor průběhu* (ProgressBar) - vizualizuje průběh delší operace počítače, jako je například stahování, přenos souborů nebo instalace
- *Formulář* (Form) - reprezentuje okno, které vytváří rozhraní aplikace
- *Číselník* (NumericUpDown) - pomocí šipek nahoru/dolů lze volit hodnota
- *Posuvník* (ScrollBar) - prvek, prostřednictvím něhož lze posouvat obsah
  - horizontální posun (HScrollBar)
  - vertikální posun (VScrollBar)
- *Stavový řádek* (StatusBar) - zobrazuje informace, obvykle se nachází na spodním okraji okna
- *Panel záložek* (TabControl)
- *Záložka* (TabPage) - jedna položka v panelu záložek
- *Textové pole* (TextBox) - slouží pro vkládání textu uživatelem
- *Panel nástrojů* (Toolbar) - jsou zde umístěny tlačítka, ikony, nabídky nebo jiné prvky
- *Položka panelu nástrojů* (ToolbarButton)
- *Posuvník* (TrackBar) - prvek s táhlem, pomocí něhož lze volit hodnotu posouváním vertikálně nebo horizontálně po liště
- *Strom* (TreeView) - prezentuje informace v hierarchickém pohledu
- *Uzel stromu* (TreeNode) - uzel ve stromovém zobrazení

A následují neviditelné prvky.

- *Dialog* (CommonDialog) - vyvolá dialogové okno pro
  - výběr barvy (ColorDialog)
  - otevření nebo uložení souboru (FileDialog)
  - výběr písma (FontDialog)
  - nastavení stránky (PageSetupDialog)
  - možnost tisku (PrintDialog)
- *Notifikační ikona* (NotifyIcon) - nastaví ikony v oznamovací oblasti
- *Časovač* (Timer) - umožňuje pracovat s časem
- *Výběr data a času* (DateTimePicker) - slouží pro volbu data a času
- *Panel* (Panel) - podobná funkce jako u skupinového rámce, navíc lze posouvat obsah

2.3 Standardní prvky systému Android

Tento operační systém obsahuje většinu prvků, které se vyskytují v systému Windows, ale přidává i nějaké jiné. Kvůli funkční podobnosti těchto prvků u nich v seznamu vypíši pouze názvy. Příloha C[7] ilustruje strukturu prvků v sytému Android. V následujícím seznamu uvedu jednotlivé prvky, ale vynechám popis u těch, které se nacházejí v systému Windows.

- *Okno* (View) - základní stavební blok pro ostatní komponenty zodpovědný za vykreslení
- *Analogové hodiny* (AnalogClock) - zobrazí analogové hodiny se dvěma ručičkami
- *Digitální hodiny* (DigitalClock) - zobrazí digitální hodiny, navíc ukazuje sekundy
- *Obrázek* (ImageView)
- *Obrázkové tlačítko* (ImageButton) - tlačítko, které obsahuje místo textu obrázek
- *Klávesnice* (KeyBoardView) - slouží pro zobrazení virtuální klávesnice
- *Indikátor průběhu* (ProgressBar)
- *Mezera* (Space) - vytvoří mezery mezi komponentami
- *Podokno* (SurfaceView) - umožňuje častější vykreslování než klasické okno
- *Podokno s OpenGL* (GLSurfaceView) - pro vykreslování OpenGL grafiky
- *Video* (VideoView) - zobrazí video soubory
- *Text* (TextView) - slouží k zobrazení textu uživateli, text nelze upravovat
- *Tlačítko* (Button)
- *Zaškrtávací tlačítko* (CheckBox)
- *Přepínač* (RadioButton)
- *Přepínací tlačítko* (Switch) - stejná funkce jako první přepínací tlačítko, ale má jiný design
- *Přepínací tlačítko* (ToggleButton) - funkčně podobné jako zaškrtávací tlačítko, navíc obsahuje přímo textovou hodnotu (např. Zapnuto/Vypnuto)
- *Časovač* (Chronometer)
- *Editovací pole* (EditText) - přidává možnost editace textu
- *Automatické doplňovací pole* (AutoCompleteTextView) - při psaní automaticky zobrazuje návrhy vět, které nahradí celý text v poli
- *Vícenásobné automatické doplňovací pole* (MultiAutoCompleteTextView) - zobrazuje návrhy částí vět
- *Textové hodiny* (TextClock) - dokáže vykreslit současné datum a/nebo čas jako formátovaný text
- *Textura* (TextureView) - používá se k zobrazení datového toku (např. videa nebo OpenGL grafiky), tváří se jako normální okno
- *Skupina* (ViewGroup) - speciální okno, které může obsahovat jiná okna

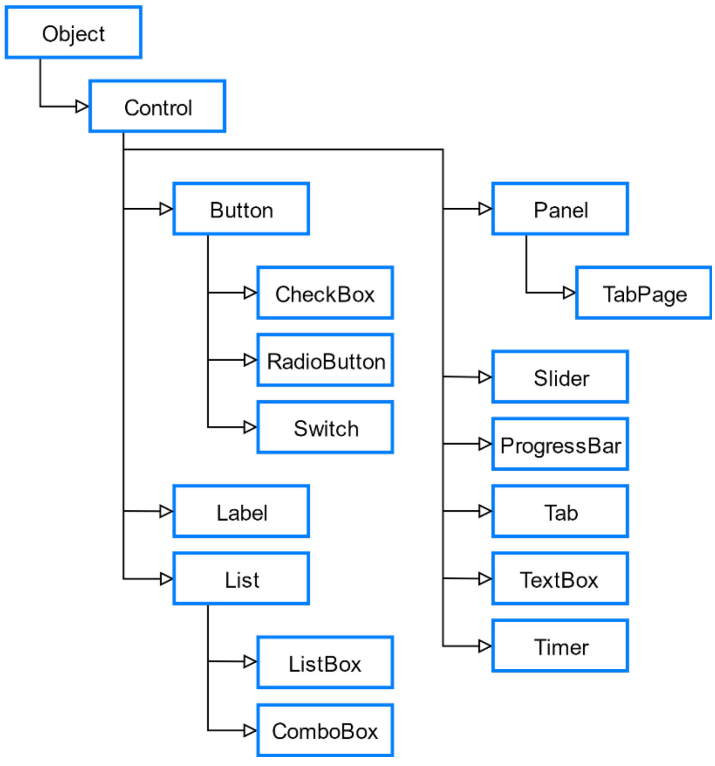
- *Absolutní rozvržení* (AbsoluteLayout) - rozvržení, které umožňuje specifikovat přesnou polohu potomků
- *Webové okno* (WebView) - okno, které zobrazuje webové stránky
- *Mřížka* (GridView)
- *Seznam* (ListView)
- *Rozbalovací seznam* (Spinner)
- *Boční seznam* (DrawerLayout) - seznam, který je možné vysunout z vertikálních okrajů okna
- *Rámcové rozvržení* (FrameLayout) - vyhradí oblast obrazovky k zobrazení jediné položky
- *Kalendář* (CalendarView) - slouží k zobrazení a výběru dat
- *Datum* (DatePicker) - pro výběr data pomocí výběrů čísla
- *Vrstva pro gesta* (GestureOverlayView) - průhledná vrstva, které umožní uživateli zadávat gesta
- *Okno pro horizontální posun* (HorizontalScrollView) - uživatel může posouvat obsah horizontálně
- *Ovládání médií* (MediaController) - okno obsahující ovládací prvky pro přehrávač
- *Okno pro vertikální posun* (ScrollView) - uživatel může posouvat obsah vertikálně
- *Okno se záložkami* (TabHost) - kontejner, který zobrazí panel se záložkami
- *Výběr času* (TimePicker) - slouží pro výběr času
- *Lineární rozvržení* (LinearLayout) - rozvrhne potomky do jednoho sloupce nebo jednoho řádku
- *Výběr čísla* (NumberPicker) - umožní uživateli vybrat číslo z předdefinovaného rozsahu
- *Vyhledávací pole* (SearchView) - rozhraní pro uživatele, pomocí kterého zadává dotazy
- *Záložky* (TabWidget) - seznam popisků záložek reprezentující jednotlivé stránky
- *Tabulkové rozvržení* (TableLayout) - uspořádá potomky do řádků a sloupců
- *Ovládání zvětšení* (ZoomControls) - zobrazí sadu ovládacích prvků pro zvětšení obsahu
- *Indikátor stránek* (PagerTitleStrip) - neinteraktivní indikátor současné, předchozí a následující stránky manažeru stránek
- *Interaktivní indikátor stránek* (PagerTitleStrip) - podobný indikátoru stránek, ale je interaktivní
- *Relativní rozvržení* (RelativeLayout) - pozice potomků je pospána vztahem ke každému dalšímu potomkovi
- *Posuvné rozložení* (SlidingPaneLayout) - horizontální posuvné rozvržení s více panely

- *Aktualizační rozvržení* (SwipeRefreshLayout) - uživatel může použít pro aktualizaci obsahu okna vertikálním gestem posunutí
- *Manažer stránek* (ViewPager) - umožňuje přesouvání vlevo a vpravo mezi stránkami dat

3 Návrh vlastní struktury prvků

3.1 Výběr prvků

Při návrhu struktury prvků, které by mohly být použity v systémech Windows a Android a které by bylo možné realizovat v prostředí Processing, jsem vybíral z těch nejpoužívanějších, se kterými se denně setkávám při používání těchto systémů. Protože musí být prvky v této struktuře použitelné v obou systémech, volil jsem z množiny prvků, které jsou pro ně společné.



Obrázek 5: Návrh struktury prvků uživatelského rozhraní

Následuje seznam s českými názvy prvků.

- Tlačítko (Button)
- Zaškrtávací tlačítko (CheckBox)
- Přepínač (RadioButton)
- Přepínací tlačítko (Switch)
- Popisek (Label)
- Seznam (ListBox)
- Rozbalovací seznam (ComboBox)
- Panel (Panel)
- Posuvník (Slider)
- Textové pole (TextBox)
- Záložka (Tab)
- Indikátor průběhu (ProgressBar)
- Časovač (Timer)

3.2 Popis prvků

V této kapitole popíši funkci jednotlivých komponent z navržené struktury. V případě viditelných prvků připojím také návrh jejich grafické podoby. Při tvorbě společné grafické stránky prvků jsem zvolil design v moderním stylu, který je vhodný pro dnešní dobu. Barevné schéma jsem přizpůsobil bílému pozadí a design je tvořen kombinacemi odstínů šedé a modré. K tvorbě vizuálního návrhu jsem použil open source vektorový editor Inkscape. V následující kapitole uvedu popis jednotlivých prvků a jejich design.

3.2.1 Tlačítko

Jeden z nejpoužívanějších grafických prvků, dovoluje uživateli spustit nějakou určitou akci.

Obrázek 6: Design tlačítka

3.2.2 Zaškrtávací tlačítko

Prvek, který umožňuje uživateli zvolit více možností z množiny voleb. Každá volba vyjadřuje Ano nebo Ne. Příklad použití - uživatel zaškrtne volbu, pokud mluví daným

jazykem (pro každý jazyk Ano/Ne). Obrázek 7 ukazuje vlevo nezaškrtnuté tlačítko, vpravo zaškrtnuté tlačítko.

Obrázek 7: Design zaškrtávacího tlačítka

3.2.3 Přepínač

Prvek, který dovolí uživateli zvolit pouze jednu z více předdefinovaných možností (vzájemně se vylučují). Když uživatel zvolí možnost, ostatní přepínače se nastaví na nezvolené. Přepínače jsou upořádány do skupin po dvou či více. Obrázek 8 ilustruje vlevo nezvolenou možnost, vpravo zvolenou možnost. Příklad použití - výběr nejvyššího dokončeného vzdělání.

Obrázek 8: Design přepínače

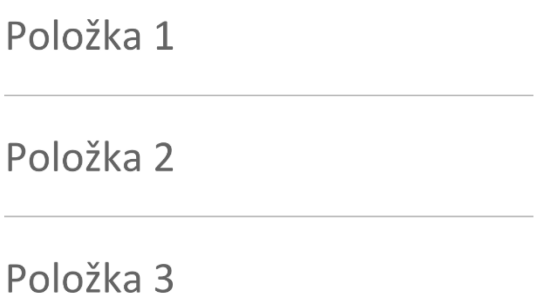
3.2.4 Přepínací tlačítko (switch)

Přepínací tlačítko má stejnou funkci jako zaškrtávací tlačítko, ale používá se k vypnutí nebo zapnutí pouze jedné volby. Obrázek 9 - vlevo vypnutý stav a vpravo zapnutý stav. Příklad použití - zapnutí nebo vypnutí připojení k bezdrátové síti.

Obrázek 9: Design přepínacího tlačítka

3.2.5 Seznam

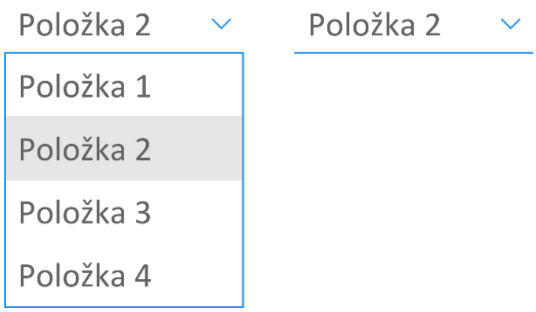
Seznam umožňuje uživateli vybrat jednu možnost ze seznamu. Používá se například v menu, kde má uživatel k dispozici výběr z více možností a kde je třeba uspořádaná struktura voleb.



Obrázek 10: Design seznamu

3.2.6 Rozbalovací seznam

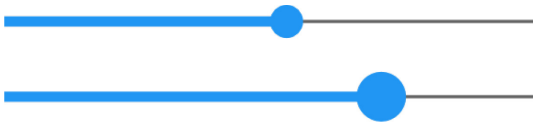
Podobný klasickému seznamu, ale pokud je neaktivní, zobrazuje jen jednu právě zvolenou možnost. Po kliknutí na tlačítko se zobrazí seznam všech hodnot, mezi kterými může uživatel vybírat. Po vybrání hodnoty seznam opět zobrazí jen zvolenou možnost. Používá se namísto přepínačů v případě, kdy je vhodnější šetřit prostorem. Obrázek 11 ukazuje vlevo rozbalený seznam se všemi možnostmi a vpravo v neaktivním stavu. Příklad použití - výběr jazyka, volba roku narození.



Obrázek 11: Design rozbalovacího seznamu

3.2.7 Posuvník (slider)

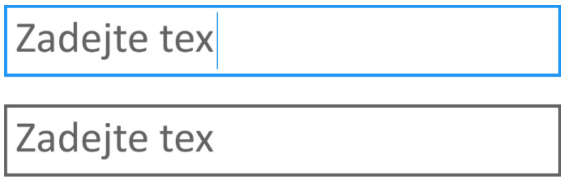
Prvek, prostřednictvím něhož může uživatel volit hodnotu pohybem táhla/indikátoru, obvykle v horizontálním směru. Zvolit hodnotu lze také výběrem konkrétního místa na posuvníku. Často se používá v kombinaci s indikátorem průběhu (např. u online videí). Obrázek 12 - nahoře neaktivní stav, dole aktivní stav při držení táhla. Příklad použití - volba hlasitosti přehrávání.



Obrázek 12: Design posuvníku

3.2.8 Textové pole

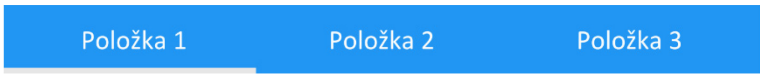
Prvek, který umožňuje uživateli zadávat textové informace, které následně program zpracuje. Textové pole obvykle při psaní zobrazuje textový kurzor (běžně vertikální čára), který indikuje současnou pozici úpravy textu. Obrázek 13 ilustruje nahoře aktivní stav při editaci, dole neaktivní stav. Příklad použití - vkládání jména a příjmení.



Obrázek 13: Design textového pole

3.2.9 Záložka

Pomocí záložek lze zobrazit více dokumentů nebo panelů v rámci jednoho okna. Při výběru záložky se zobrazí odpovídající dokument. Vhodné použití v případě, pokud potřebujeme zobrazit velké množství dat. Tato data lze roztřídit podle kontextu a zobrazovat interakcí se záložkami. Obrázek 14 zobrazuje vybranou první záložku a v okně se zobrazí odpovídající dokument. Příklad použití - zobrazení informací o datových připojeních (každé na jiné záložce).



Obrázek 14: Design záložek

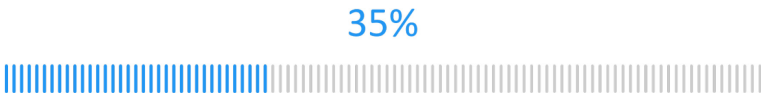
3.2.10 Popisek

Popisek zobrazuje text v aplikaci. Obvykle je statický, neobsahuje žádnou interaktivitu. Je používán k identifikaci textových polí nebo jiných prvků. Příklad použití - popis prvků ve formuláři.

Obrázek 15: Design popisku

3.2.11 Indikátor průběhu

Prvek, který se používá k vizualizaci průběhu nějaké delší operace počítače. Obvykle zobrazuje procentuální průběh této operace. Příklad použití – grafické znázornění průběhu stahování, posílání souborů nebo instalace programů.



Obrázek 16: Design indikátoru průběhu

3.2.12 Panel

Tato komponenta patří do skupiny neviditelných prvků. Jedná se o prvek, který slouží jako kontejner pro ostatní objekty. Při jejich umístění na panel se změní jejich poloha a ta je poté závislá na umístění tohoto panelu. Při změně jeho pozice se posunou i ostatní komponenty na panelu.

3.2.13 Časovač

Jedná se o další neviditelný prvek. Jak už vyplývá z názvu této komponenty, její funkcí je měření času, na jehož základě se spustí nastavená úloha. Tuto úlohu lze vykonávat cyklicky nebo ji lze spustit pouze jednou.

4 Realizace vybraných prvků v prostředí Processing

4.1 Části tvorby prvků

K vytvoření vizuální podoby prvků jsem použil grafické objekty prostředí Processing (obdélník, čára, elipsa, text). Poté jsem pomocí funkcí dostupných v tomto prostředí a vlastních metod vytvořil odpovídající chování jednotlivých prvků. Dále popíši hlavní části jejich realizace.

4.1.1 Reakce na vstup od uživatele

Zde využívám metody prostředí Processing, pomocí kterých zjišťuji aktuální souřadnice kurzoru při kliknutí myší nebo bod na displeji, kam uživatel klepl. Pomocí této informace určuji, zda uživatel klikl na komponentu nebo ne. V případě textového pole kontroluji zadání znaku.

Jedná se o metody *mouseEvent* a *keyEvent*, které reprezentují události myši a klávesnice. Ve funkci *mouseEvent* pracuji s informacemi o pozici kurzoru a kliknutí myší. Pomocí metody *keyEvent* zaznamenávám stisknutí kláves na klávesnici.

4.1.2 Události prvků

Pokud uživatel na daný prvek klikl nebo klepl, spustí se reakce formou zavolání události. Tato událost je reprezentována objektem třídy *GEvent*, který v sobě nese informaci o prvku, jenž tuto událost vyvolal.

V prostředí Processing, kde si uživatel vytváří svou aplikaci, se vyvolání události projeví zavoláním metody s názvem *action*. Do těla této metody poté uživatel napíše svůj kód, který se vykoná při interakci s komponentou. K rozlišení jednotlivých prvků může uživatel použít metodu *getSource* třídy *GEvent*. Ta vrátí informaci o prvku, který danou událost vyvolal.

4.1.3 Líné vykreslování

Princip líného vykreslování spočívá v tom, že prvky se nepřekreslují stále dokola, ale pouze v případě, kdy dojde k vizuální změně jejich vlastností, např. při kliknutí na přepínací tlačítko. Pro použití líného vykreslování jsem se rozhodl z důvodu šetření systémových prostředků, protože aplikace vytvořené v prostředí Processing se ve výchozím nastavení snímkové frekvence překreslují 60krát za vteřinu.

Pro realizaci funkce líného vykreslování obsahuje každý viditelný prvek proměnnou *isValid*, která udává, zda je daný prvek tzv. validní neboli platný. Pokud je prvek platný, je vykreslený správně. Jestliže platný není, jeho zobrazení neodpovídá aktuálnímu stavu, a proto je nutné jej vykreslit znovu.

Samotná kontrola změny vlastností prvků se provádí ve funkcích, které slouží pro nastavení proměnných ovlivňujících tyto vlastnosti. Pokud nastane změna, dojde k překreslení komponenty. Pro účel hromadného překreslení jsem vytvořil třídu *Validator*. Tato třída obsahuje pole všech viditelných prvků v aplikaci a metodu *invalidate*, pomocí které lze tyto prvky najednou zneplatnit, tj. nastavit jejich proměnnou *isValid* na nepravdu.

4.2 Realizace vybraných prvků

Z navržených prvků jsem realizoval množinu komponent, které umožní vytvářet jednoduché a snadno ovladatelné multiplatformní aplikace. Tato množina obsahuje tyto prvky:

- tlačítko
- zaškrtávací tlačítko
- přepínač
- přepínací tlačítko
- posuvník
- textové pole

- panel
- časovač

Při realizaci těchto prvků jsem vycházel ze struktury uvedené v kapitole 3.1. Každý prvek je na nejvyšší úrovni potomkem třídy *Control*, která obsahuje všechny společné atributy a metody, např. informace o parametrech prvků nebo metodu pro vyvolání událostí.

Struktura tříd reprezentujících jednotlivé komponenty je rozdělena na tři hlavní části. Těmi jsou konstruktor, metody událostí (*mouseEvent*, *keyEvent*) a metoda *draw*. V konstruktoru se mimo nastavení atributů volají funkce pro zaregistrování událostí a metody pro přidání prvku do pole třídy *Validator*. Metody událostí jsem již popisoval, jsou nezbytné pro funkčnost prvků. Poslední zmíněná funkce je důležitá pro samotné vykreslení, jelikož kód, který obsahuje, se vykoná ve stejně nazvané metodě v uživatelské aplikaci, jenž slouží právě pro vykreslení objektů. Dále najdete popis tvorby jednotlivých prvků.

4.2.1 Tlačítko

Tento prvek zastupuje třída *Button*. Tato třída je ze všech ostatních, které reprezentují viditelné prvky, nejjednodušší. Kontrola stisku tlačítka probíhá v metodě *mouseEvent*. Zde k tomuto účelu slouží kombinace metod *isMouseOver* a *getAction*. První funkce určuje, zda se kurzor nachází nad komponentou. Druhá poskytuje informaci o stisku tlačítka myši. K vyvolání události tedy dojde při splnění podmínek v tomto pořadí:

- uživatel stiskl tlačítko myši
- funkce *isMouseOver* vrátila hodnotu pravda
- uživatel pustil tlačítko myši

Třída tlačítka obsahuje také metodu *draw*, jejíž funkci jsem již popisoval.

4.2.2 Zaškrtávací tlačítko, přepínač a přepínací tlačítko

Jelikož jsou tyto komponenty funkčně podobné tlačítku, tak jsou třídy *CheckBox*, *Switch* a *RadioButton* potomci třídy *Button*. V jejich těle se však změní metody *mouseEvent* a *draw*, kde se projevují funkční a grafické odlišnosti těchto prvků od tlačítka.

Největší rozdíl ve funkci oproti tlačítku najdeme ve třídě přepínače. Tato komponenta ke své činnosti využívá třídu *RadioGroup*, která představuje skupinu přepínačů. V těle třídy *RadioButton* proto najdeme metody, které spolupracují právě s touto třídou.

4.2.3 Posuvník

Pro tento prvek jsem vytvořil třídu *Slider*. Posuvník je oproti ostatním prvkům specifický v tom, že je k ovládání jeho táhla potřebný pohyb myši. Při tomto pohybu dochází k uzamknutí ovládání po dobu držení tlačítka myši. Při jeho uvolnění dojde k odemknutí a táhlo již nelze ovládat.

V této třídě najdeme především funkce, jenž pracují s hodnotou, kterou vyjadřuje pozice posuvníku. Například funkce *getPercent* poskytuje tuto hodnotu vyjádřenou v procentech. Dalším rozdílem je odlišná implementace metody *isMouseOver*, protože obrys tohoto prvku netvoří obdélník.

4.2.4 Textové pole

Mezi realizované viditelné prvky patří také textové pole, jehož funkci obstarává třída *TextBox*. Tato komponenta se od ostatních liší v tom, že kromě myši využívá i vstup z klávesnice. Proto je zde využita navíc metoda *keyEvent*, která umožňuje právě práci s klávesnicí.

Než však uživatel začne psát, musí kliknout do pole pro text. Po této akci se komponenta nastaví jako aktivní, tj. připravená pro příjem znaků z klávesnice. Pro nastavení aktivního módu je zde funkce *setFocus*. Při kliknutí do textového pole záleží i na místě v poli, kam uživatel kliknul. Tato informace totiž slouží pro umístění editačního kurzoru na pozici, kam se má vložit znak.

Jednou z funkcí této třídy je proto zobrazení tohoto kurzoru, který bliká. Mezi další vlastnosti patří posunutí kurzoru pomocí kurzorových šipek. Šipka vlevo/vpravo jej přesune o jeden znak vlevo/vpravo, šipka nahoru na začátek textu a šipka dolů na konec textu. Nechybí zde ani možnost mazání znaků pomocí kláves Backspace a Delete.

Za zaznamenání stisku těchto i ostatních kláves je zodpovědná metoda *keyEvent*, ve které se podle kódu stisknuté klávesy určí, zda se jedná o funkční klávesu nebo o klávesu reprezentující tisknutelný znak. Protože ale tyto kódy nejsou v systémech Windows a Android shodné a zaznamenávání kláves je odlišné, vytvořil jsem funkci *checkAndroid* k určení, zda je aplikace spuštěna na mobilní platformě či nikoliv. Pokud tedy uživatel spustil aplikaci v systému Android, jsou potřeba také metody na vysunutí a zasunutí virtuální klávesnice. Při přepnutí textového pole do aktivního režimu dojde k vysunutí klávesnice, v neaktivním režimu se zasune zpět.

Uživatel mimo psaní textu do pole může také využít funkci *setValue* pro nastavení obsahu pole. Toto je užitečné zejména při používání textového pole pro zobrazení textu.

4.2.5 Panel

Funkce panelu, jenž je reprezentován třídou *Panel*, je v seskupení prvků. Tyto prvky se umístí na panel, což znamená, že se přepočítají souřadnice jejich polohy. Dále se vloží do pole prvků, které panel obsahuje.

Jelikož se jedná o neviditelný prvek, není potřeba implementovat metody *mouseEvent* a *draw*. Pro potřeby překreslení prvků, které leží na panelu, je zde funkce *setValid* upravena tak, aby zneplatnila všechny tyto prvky.

4.2.6 Časovač

Druhou realizovanou komponentou, která patří do skupiny neviditelných prvků, je časovač. Třída *Timer*, zastupující tento prvek, pracuje s časem. K tomu používá metodu *millis* obsaženou v třídě PApplet, která měří čas od spuštění aplikace v milisekundách.

Při vytváření časovače uživatel uvede interval, v jakém intervalu bude vyvolávat událost. Tento časový údaj poté využívá metoda *draw* pro porovnání s uplynulým časem od poslední události. Pro ovládání časovače může uživatel použít funkce *start* a *stop*, určené pro jeho spuštění a zastavení.

4.3 Vytvoření knihovny

K vytvoření knihovny jsem použil prostředí Eclipse a šablonu pro tvorbu knihovny v tomto prostředí. Do této šablony jsem vložil soubory tříd, které reprezentují jednotlivé prvky. Protože třídy v knihovně jsou psané v jazyce Java, který neobsahuje uvedené grafické objekty jazyka Processing, bylo třeba upravit jejich kód. Hlavním bodem této úpravy bylo importování balíku PApplet, který umožní komunikaci se sketchem v prostředí Processing a také použití jeho grafických objektů a metod.

Dále bylo potřeba upravit viditelnost konstruktorů a metod, které jsou v jazyce Processing pouze soukromé, tak, aby byly dostupné v rámci knihovny. Po úpravě jsem do šablony vložil sketche s příklady použití jednotlivých prvků. Nakonec jsem celý projekt se šablonou zkompiloval pomocí nástroje Ant.

Pro použití knihovny v prostředí Processing je poté nutné vložit vytvořenou knihovnu do adresáře s knihovnami, jenž toto prostředí používá. Po opětovném spuštění prostředí Processing je již knihovna nainstalovaná a v nabídce *Examples* lze najít přiložené příklady s použitím vytvořených prvků.

4.4 Příklady použití prvků

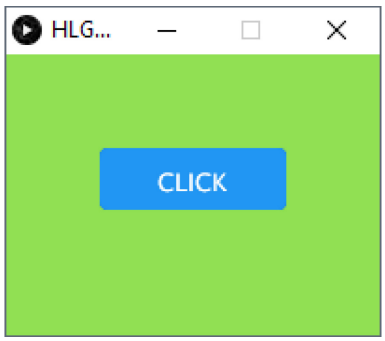
V této kapitole uvedu příklady použití jednotlivých prvků, které jsou obsažené ve vytvořené knihovně. Kromě popisu každého příkladu přiložím také ukázku kódu v prostředí Processing a ilustraci odpovídající aplikace.

4.4.1 Tlačítko

Tento příklad ukazuje funkci klasického tlačítka, tj. po kliknutí na tlačítko se změní barva pozadí na náhodnou hodnotu.

```
import HLGUI.*;
Button b;
// nastavení parametrů aplikace a vytvoření tlačítka
void setup(){
  size (200, 150);
  b = new Button(this, 50, 50, 100, 33,
  "CLICK");
}
void draw(){}
// metoda, která se zavolá při kliknutí na tlačítko
void action(GEvent e){
  if (e.getSource() == b) {
    background(random(255), random(255), random(255));
  }
}
```

Zdrojový kód 6: Příklad použití tlačítka v knihovně



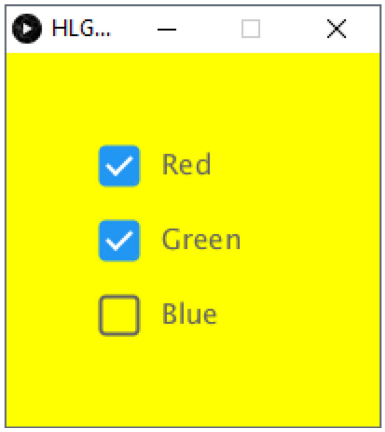
Obrázek 17: Vzhled aplikace z příkladu použití tlačítka

4.4.2 Zaškrtávací tlačítko

V této ukázce jsou umístěny tří zaškrtávací tlačítka, které mění barvu pozadí. Každé reprezentuje jednu barvu z barevného modelu RGB, tj. červenou, zelenou a modrou. V neaktivním stavu je hodnota odpovídající barvy nastavena na 0, v aktivním stavu má hodnotu 255. Při zaškrtnutí více tlačítek se pozadí obarví kombinací zvolených barev.

```
import HLGUI.*;
CheckBox c1, c2, c3;
// nastavení parametrů aplikace a vytvoření zaškrtávacích tlačítek
void setup(){
  size (200, 200);
  c1 = new CheckBox(this, 50, 50, 20, "Red");
  c2 = new CheckBox(this, 50, 90, 20, "Green");
  c3 = new CheckBox(this, 50, 130, 20, "Blue");
}
void draw(){}
// metoda, která se zavolá při kliknutí na zaškrtávací tlačítko
void action(GEvent e){
  if (!c1.isChecked() && !c2.isChecked()
      && !c3.isChecked()) background(255);
  background(255*(c1.isChecked() ? 1 : 0),
  255*(c2.isChecked() ? 1 :  0), 255*(c3.isChecked() ? 1 : 0));
}
```

Zdrojový kód 7: Příklad použití zaškrtávacího tlačítka v knihovně



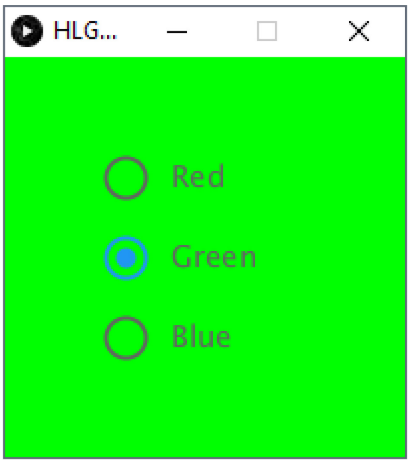
Obrázek 18: Vzhled aplikace z příkladu použití zaškrtávacího tlačítka

4.4.3 Přepínač

Podobný příklad jako ukázka zaškrtávacího tlačítka, pouze jsou přepínače umístěny do skupiny, takže lze v jednu chvíli zvolit pouze jeden přepínač.

```
import HLGUI.*;
RadioButton r1, r2, r3; RadioGroup rg;
// nastavení parametrů aplikace a vytvoření přepínačů
void setup(){
  size (200, 200);
  r1 = new RadioButton(this, 50, 50, 20, "Red");
  r2 = new RadioButton(this, 50, 90, 20, "Green");
  r3 = new RadioButton(this, 50, 130, 20, "Blue");
  rg = new RadioGroup();
  rg.addButton(r1); rg.addButton(r2); rg.addButton(r3);
  rg.select(r2);
}
void draw(){}
// metoda, která se zavolá při kliknutí na přepínač
void action(GEvent e){
  switch (rg.getIndex()){
    case 0 : background(255,0, 0); break;
    case 1 : background(0,255,0); break;
    case 2 : background(0, 0, 255); break;
    default: background(255);
  }
}
```

Zdrojový kód 8: Příklad použití přepínače v knihovně



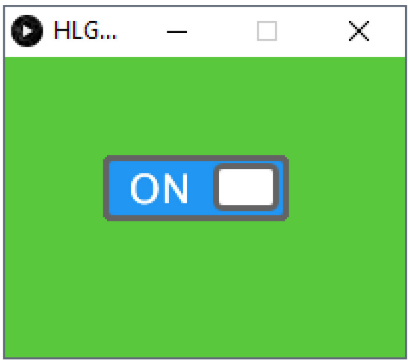
Obrázek 19: Vzhled aplikace z příkladu použití přepínače

4.4.4 Přepínací tlačítko

Příklad ukazuje funkci přepínacího tlačítka, které ovládá změnu barvy pozadí na zelenou. V aktivním stavu je tedy barva pozadí zelená, v neaktivním je pozadí bílé.

```
import HLGUI.*;
Switch s;
// nastavení parametrů aplikace a vytvoření přepínacího tlačítka
void setup(){
  size (200, 150);
  s = new Switch(this, 50, 50, 30);
}
void draw(){}
// metoda, která se zavolá při kliknutí na přepínací tlačítko
void action(GEvent e){
  if (s.getState()) background(0, 100, 0);
  else background(255);
}
```

Zdrojový kód 9: Příklad použití přepínacího tlačítka v knihovně



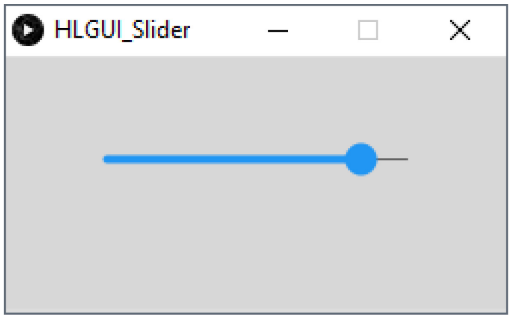
Obrázek 20: Vzhled aplikace z příkladu použití přepínacího tlačítka

4.4.5 Posuvník

Tato ukázka zobrazí posuvník, jehož hodnota udává podíl bílé barvy v barvě pozadí. Pokud je táhlo posuvníku zcela vpravo, hodnota posuvníku odpovídá 100% a pozadí je tedy bílé. Při posunutí táhla v levém směru se barva postupně změní z bílé přes odstíny šedé až po černou.

```
import HLGUI.*;
Slider s;
// nastavení parametrů aplikace a vytvoření posuvníku
void setup(){
  size (250, 125);
  s = new Slider(this, 50, 50, 150, 75);
}
void draw(){}
// metoda, která se zavolá při změně pozice táhla posuvníku
void action(GEvent e){
  if (e.getSource() == s) background(255*s.getPercent());
}
```

Zdrojový kód 10: Příklad použití posuvníku v knihovně



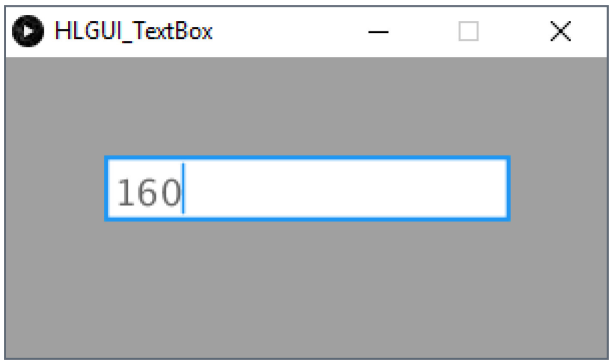
Obrázek 21: Vzhled aplikace z příkladu použití posuvníku

4.4.6 Textové pole

Na tomto příkladu je demonstrována funkce textového pole. Pokud uživatel do pole zadá celočíselnou hodnotu v intervalu 0 až 255, nastaví se barva pozadí na odpovídající hodnotu od černé po bílou. Pokud je vloženo cokoliv jiného, barva se nezmění.

```
import HLGUI.*;
TextBox t;
// nastavení parametrů aplikace a vytvoření textového pole
void setup(){
  size (300, 150);
  t = new TextBox(this, 50, 50, 200, 30, "230");
  background(int(t.getValue()));
}
void draw(){}
// metoda, která se zavolá při změně hodnoty v textovém poli
void action(GEvent e){
  if (e.getSource() == t) if (int(t.getValue()) >= 0
    && int(t.getValue()) <= 255) background(int(t.getValue()));
}
```

Zdrojový kód 11: Příklad použití textového pole v knihovně



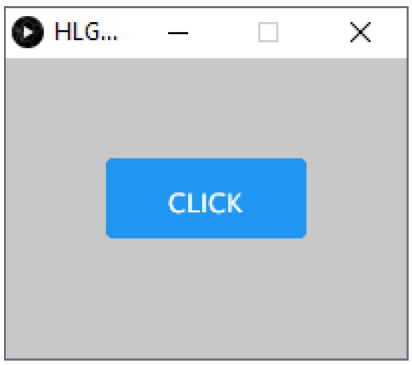
Obrázek 22: Vzhled aplikace z příkladu použití textového pole

4.4.7 Panel

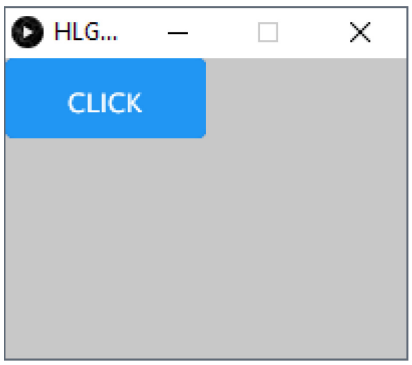
V této ukázce je umístěno tlačítko, které je umístěno do levého horního okraje aplikace. Protože ale leží na panelu, který je od levého a horního okraje vzdálený sto obrazových bodů, je stejně daleko i tlačítko. Při klinutí na tlačítko se toto tlačítko odebere z panelu nebo se na něj vloží.

```
import HLGUI.*;
Button b; Panel p;
// nastavení parametrů aplikace, vytvoření panelu a tlačítka
void setup(){
  size (200, 150);
  b = new Button(this, 0, 0, 100, 40, "CLICK");
  p = new Panel(this, 50, 50, 100, 100);
  p.add(b);
}
void draw(){}
// metoda, která se zavolá při kliknutí na tlačítko
void action(GEvent e){
  background(200);
  if (e.getSource() == b) if (p.contains(b)) p.remove(b);
  else p.add(b);
}}
```

Zdrojový kód 12: Příklad použití panelu v knihovně



Obrázek 23: Vzhled aplikace z příkladu použití textového pole – tlačítko je na panelu



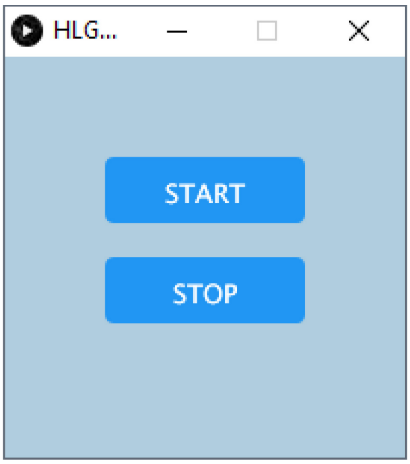
Obrázek 24: Vzhled aplikace z příkladu použití textového pole – tlačítko není na panelu

4.4.8 Časovač

Příklad obsahuje dvě tlačítka a časovač, jehož interval je nastaven na 1000 milisekund. Tlačítkem *start* uživatel spustí časovač. Po spuštění se akce provádí cyklicky po uplynutí nastavené doby. Zde se každou vteřinu změní barva pozadí na náhodnou. Po stisku tlačítka *stop* se časovač zastaví.

```
import HLGUI.*;
Button b1, b2;
Timer t;
// nastavení parametrů aplikace a vytvoření časovače
void setup(){
  size (200, 200);
  b1 = new Button(this, 50, 50, 100, 33, "START");
  b2 = new Button(this, 50, 100, 100, 33, "STOP");
  t = new Timer(this, 1000);
}
void draw(){}
// metody, které se zavolají po každém cyklu časovače
   a stisknutí tlačítka
void action(GEvent e){
  if (e.getSource() == b1) t.start();
  else if (e.getSource() == b2) t.stop();
  else if (e.getSource() == t){
    background(random(255), random(255), random(255));
  }
}
```

Zdrojový kód 13: Příklad použití časovače v knihovně



Obrázek 25: Vzhled aplikace z příkladu použití časovače

Závěr

V této práci jsem se seznámil s užitečným nástrojem pro multiplatformní vývoj, pomocí kterého mohou uživatelé vytvářet aplikace s vlastním designem a funkcemi, které jiná prostředí neobsahují. K tomu jim pomáhají integrované i uživatelské knihovny grafických prvků. Právě uživatelské knihovny se týkaly mé práce. Po vyzkoušení dostupných knihoven s prvky uživatelského rozhraní jsem uvedl jejich popis spolu s příkladem vytvoření prvku a jeho obrázku.

Součástí návrhu bylo prozkoumání prvků v systémech Windows a Android. V práci jsem toto ukázal pomocí ilustrací struktury a popisu jednotlivých prvků. Dále jsem navrhl jednoduchou strukturu nejpoužívanějších prvků společných pro oba systémy, jejíž ilustrace je v práci obsažena. Grafický design prvků jsem pojal v moderním stylu, je použitelný pro obě platformy. Každý prvek jsem popsal a připojil obrázek. V práci je také uvedena realizace vybraných komponent, které umožní vytvářet jednoduché multiplatformní aplikace, a následné vytvoření knihovny pro prostředí Processing. Poté jsem vytvořil příklady použití prvků v této knihovně, jež jsem v práci také popsal.

V budoucnu by vývoj mohl směřovat k rozšíření o další, méně používané prvky. Dále by bylo možné vytvořit grafický designér k usnadnění umístění prvků a nastavení jejich vlastností.

Použitá literatura

[1] Processing (programming language). In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2017-03-02]. Dostupné z: https://en.wikipedia.org/wiki/Processing_(programming_language)

[2] Widget (GUI). In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2017-03-04]. Dostupné z: https://en.wikipedia.org/wiki/Widget_(GUI)

[3] ControlP5. *GitHub* [online]. c2015 [cit. 2017-03-05]. Dostupné z: https://github.com/sojamo/controlp5

[4] G4P (GUI for processing). *Quarks place* [online]. [cit. 2017-03-05]. Dostupné z: http://www.lagers.org.uk/g4p/

[5] Libraries. *Processing* [online]. [cit. 2017-03-09]. Dostupné z: https://processing.org/reference/libraries/

[6] Windows Forms Component Hierarchy. In: *Microsoft* [online]. Redmond (WA): Microsoft, c2017 [cit. 2017-03-04]. Dostupné z: https://msdn.microsoft.com/en-us/library/8w7ed3ba(v=vs.71).aspx

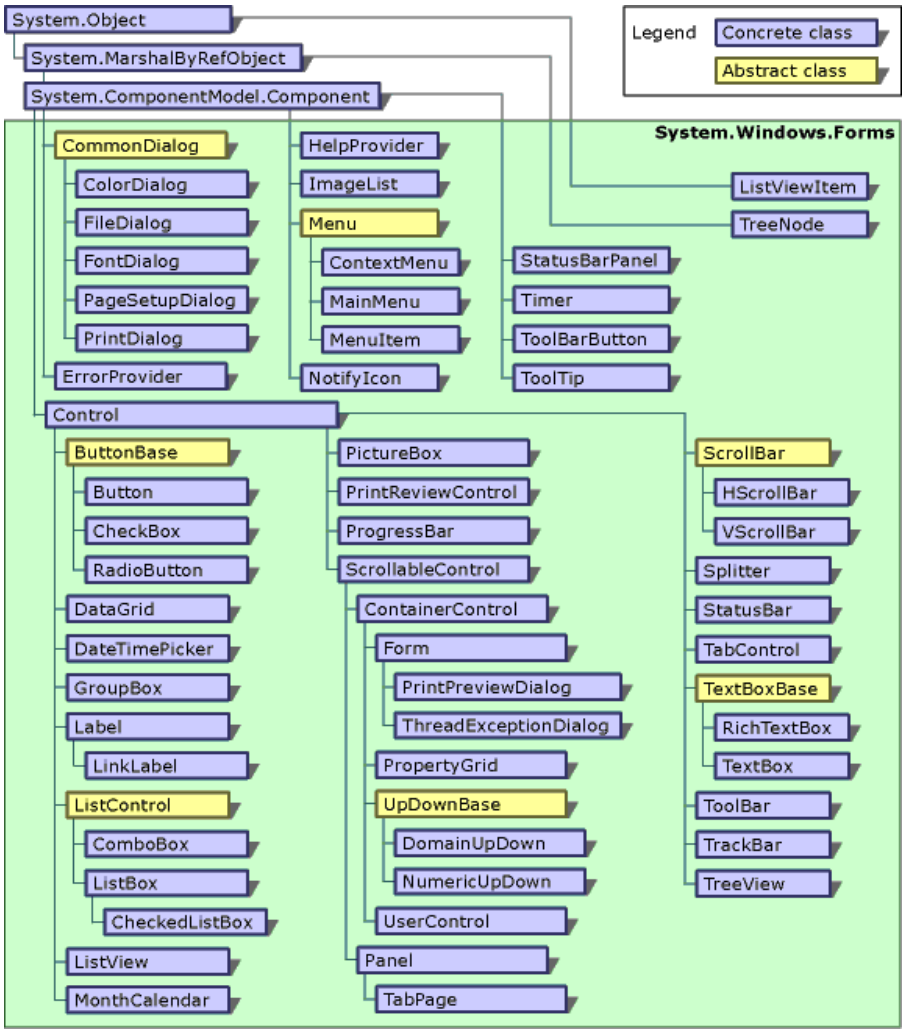
[7] THORMÄHLEN, Thorsten. Graphics Programming GUIs with Java (Android). In: *Philipps-Universität Marburg* [online]. Marburg (DE): Philipps-Universität Marburg, 2016 [cit. 2017-03-04]. Dostupné z: http://www.mathematik.uni-marburg.de/~thormae/lectures/graphics1/graphics_2_3_eng_web.html

A Obsah přiloženého CD

- text bakalářské práce ve formátu PDF
  - Bakalarska_prace_Lukas_Hradecky.pdf

- archiv s vytvořenou knihovnou
  - HLGUI.zip

B Struktura prvků uživatelského rozhraní v platformě .NET



C Struktura prvků uživatelského rozhraní v systému Android