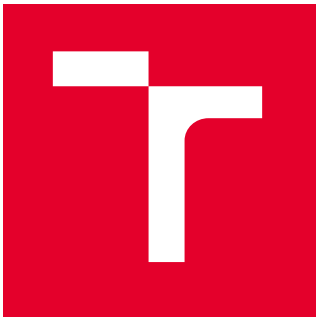


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ

DEPARTMENT OF TELECOMMUNICATIONS

EXPERIMENTÁLNÍ SOFTWAROVÝ NÁSTROJ ZAMĚŘENÝ NA ZVUKOVOU TVORBU PROSTŘEDNICTVÍM SEKVENCÍ

EXPERIMENTAL SOFTWARE TOOL FOCUSED ON SOUND CREATION THROUGH SEQUENCES

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — Jan Trubačík

AUTHOR

VEDOUCÍ PRÁCE — MgA. Michal Indrák, Ph.D.

SUPERVISOR

BRNO 2024

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Audio inženýrství**
specializace Zvuková produkce a nahrávání
Ústav telekomunikací

Student: Jan Trubačík ***ID:*** 226484
Ročník: 3 ***Akademický rok:*** 2023/24

NÁZEV TÉMATU:

Experimentální softwarový nástroj zaměřený na zvukovou tvorbu prostřednictvím sekvencí

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je funkční software, který bude schopen vytvářet zvukové plochy z generovaných sekvencí. Sekvence bude možné vkládat různými způsoby a Součástí nástroje bude i syntéza zvuku s využitím těchto sekvencí, kontrola jeho průběhu a jeho detailní nastavení.

V rámci semestrální práce student nástroj kompletně navrhne a vytvoří ukázky programového řešení jednotlivých částí.V navazující bakalářské práce potom realizuje plně funkční program.

Kontakt: Michal Indrák na HF JAMU po předchozí domluvě e-mailem (indrak@jamu.cz)

DOPORUČENÁ LITERATURA:

[1] RUSS, Martin. Soundsynthesis and sampling. Oxford: FocalPress, 1996. Music technology series. ISBN 0-240-51429-7.
[2] TheCsoundbook: perspectives in software synthesis, sound design, signalprocessing, and programming. Editor Richard BOULANGER. Cambridge: MIT Press, c2000. ISBN 0262522616.

Termín zadání: 5.2.2024 ***Termín odevzdání:*** 28.5.2024

Vedoucí práce: MgA. Michal Indrák, Ph.D.

doc. Ing. Jiří Schimmel, Ph.D.
předseda rady studijního programu

ABSTRAKT

Cílem bakalářské práce je sestavení experimentálního softwarového nástroje určeného ke zvukové tvorbě prostřednictvím sekvencí. Realizace je provedena v grafickém programovacím prostředí Pure Data. Nástroj dekóduje textové řetězce a převádí je v reálném čase na sekvence MIDI zpráv. Text práce představuje klíčové teoretické poznatky a popisuje obsluhu nástroje a jeho architekturu.

KLÍČOVÁ SLOVA

softwarový hudební nástroj, MIDI, Pure Data, sekvencer, experimentální hudba

ABSTRACT

The goal of this bachelor's thesis is the creation of an experimental musical software tool, which is designed for generating music through the use of sequences. The tool was created using Pure Data, a visual programming environment. It decodes strings of plain text and converts them to sequences of MIDI messages in real time. The text of the thesis illustrates a theoretical background and the tool's structure and way of use.

KEYWORDS

musical software, MIDI, Pure Data, music sequencer, experimental music

Vysázeno pomocí balíčku `thesis` verze 4.09; `https://latex.fekt.vut.cz/`

TRUBAČÍK, Jan. *Experimentální softwarový nástroj zaměřený na zvukovou tvorbu prostřednictvím sekvencí*. Bakalářská práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2024. Vedoucí práce: MgA. Michal Indrák, Ph.D.

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Jan Trubačík
VUT ID autora:	226484
Typ práce:	Bakalářská práce
Akademický rok:	2023/24
Téma závěrečné práce:	Experimentální softwarový nástroj zaměřený na zvukovou tvorbu prostřednictvím sekvencí

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora*

*Autor podepisuje pouze v tištěné verzi.

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu bakalářské práce panu MgA. Michalu Indrákovi, Ph.D. za odborné vedení, konzultace a inspirativní návrhy k postupu. Dále Rebece Koryčanské za konzultaci argumentace.

Obsah

Seznam obrázků

Úvod

Při tvorbě experimentálních hudebních kompozic umělec často využívá prostředků, které generují hudebně interpretovatelný výstup z uživatelem zadaných dat. Takovými daty mohou být například matematické řady nebo rastrové obrázky. Umělec tak namísto tradičního způsobu kompozice dosahuje esteticky žádoucích výsledků převodem informace ryze nehudební na informaci hudební.

Softwarových nástrojů určených k tomuto účelu existuje celá řada. Jejich základními odlišujícími faktory jsou podoba vstupu (jakou infomraci zpracovávají a jakým způsobem ji uživatel zadává) a podoba výstupu (charakter hudební informace a její datový formát).

Cílem této bakalářské práce bylo vytvořit softwarový nástroj, který generuje tóny z textových sekvencí. Na jeho vstupu je textový soubor nebo uživatelem zadaná posloupnost znaků. Výstupem je sekvence zpráv MIDI protokolu přehrávaných v reálném čase. Uživatel stanovuje pravidla, podle kterých je každý znak převeden na hudební tón a ovládá délku jednotlivých kroků sekvenceru. K řešení bylo využito grafické programovací prostředí Pure Data. Výsledným nástrojem je patch pro Pure Data obsahující několik subpatchů.

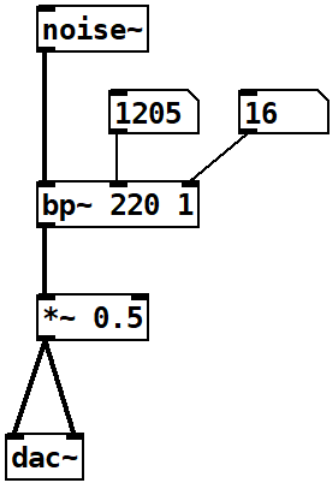
První kapitolu práce tvoří seznámení s jazykem Pure Data, protokolem MIDI a dalšími teoretickými poznatky. Druhá kapitola popisuje návrh, využití a funkce nástroje. Třetí kapitola se zabývá architekturou patche a demonstruje programové řešení jednotlivých subpatchů. V poslední kapitole jsou rozebrána omezení nástroje a možnosti jeho budoucího rozšíření.

..

1 Teoretická část

1.1 Pure Data

Pure Data (Pd) je open source grafické programovací prostředí určené k tvorbě elektronické hudby. Programování v Pd spočívá v editaci tzv. *patchů*, což jsou soubory s příponou `.pd`. Každý takový soubor má podobu plátna, na které uživatel umísťuje jednotlivé grafické prvky. Mezi prvky se signály přenášejí po čarách, které symbolizují propojení fyzickými kabely (viz Obr. 1.1). Obsluha patche uživatelem je provozována v základním režimu zobrazení (*execute* mód). Umísťování prvků a jejich propojování je možné provádět v režimu editace (*edit* mód).



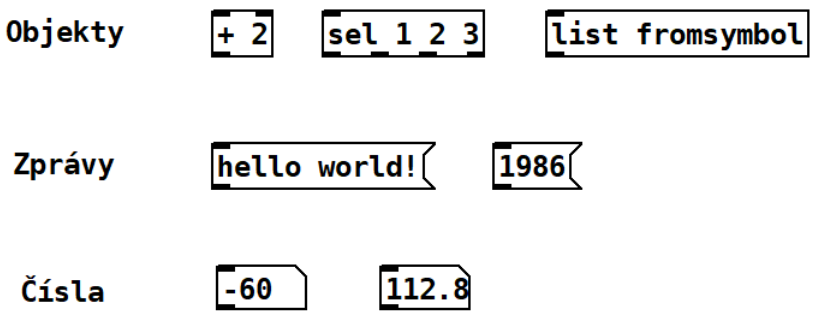
Obr. 1.1: Příklad propojení několika objektů v Pure Data

Hudební nástroje vytvořené v prostředí Pd zpracovávají data v reálném čase. V patchích mohou být prováděny změny v průběhu jejich používání a není třeba naprogramovaný software napřed sestavovat. [1]

Patche vytvořené v Pd jsou volně šiřitelné, což platí i pro zhotovený nástroj. Jeho jednotlivé části, zejména patch představující převodník znaků, lze snadno převzít a využít v budoucích Pd projektech.

1.1.1 Základní prvky

Nejběžnějšími prvky v Pd jsou **objekty** (*objects*), **zprávy** (*messages*) a **čísla** (*numbers)*. Jejich grafická podoba je vidět na obrázku 1.2. Vstupy a výstupy jsou reprezentovány drobnými černými obdélníky podél okrajů prvků. Pro každý prvek v Pd platí, že jeho vstupy jsou uspořádany podél jeho vrchní strany a jeho výstupy podél strany spodní.



Obr. 1.2: Základní prvky v Pd

Prvek typu **objekt** provádí určitou operaci. Jaká operace bude provedena je dáno typem objektu a jeho argumenty. Významné objekty `select`, `send` a `trigger` jsou stručně popsány v podkapitole 1.1.3. V rámci výkladu v kapitole 3 pak budou zmíněny důležité vlastnosti dalších použitých objektů. Seznam všech objektů a popis jejich funkce je dostupný v `online databázi`.

Prvek typu **zpráva** obsahuje řetězec znaků. Při kliknutí myší na zprávu v *execute* módu vyšle ze svého výstupu zprávu obsahující tento řetězec. Obsah prvku zprávy je uložen v momentě ukládání patche a je tedy při opětovném otevření zachován. Je nutné podotknout, že všechny řetězce znaků v Pd jsou nazývány „zprávami“, bez ohledu na to, jaký prvek je vysílá. Prvek typu zpráva může uchovávat jakýkoliv takový řetězec (včetně čísel).

Prvek typu **číslo** obsahuje číslo s plovoucí řádovou čárkou. Jeho hodnota není při zavření patche uchována a při opětovném otevření je vždy nulová. Hodnotu v prvku číslo lze narozdíl od hodnoty v prvku zprávy upravit i v *execute* módu.

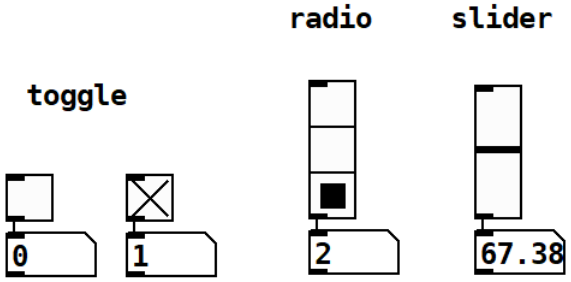
Jak je patrné již z obrázku 1.1, Pd zobrazuje propojení mezi prvky čarami ve dvou různých tloušťkách. Tlusté čáry symbolizují přenos digitálního audio signálu, tenké čáry přenos zpráv. Objekty, které zpracovávají audio signál mají navíc název zakončený vlnovkou. Vypracovaný nástroj využívá pouze spojů a objektů přenášejících zprávy.

1.1.2 GUI prvky

Velmi důležitou zprávou v Pd je textový řetězec „`bang`“. Pro přijímající prvek znamená pobídku k činnosti. Prvky typu číslo a zpráva po přijetí zprávy „`bang`“ vyšlou na svůj výstup uloženou hodnotu. Většina objektů rovněž reaguje na přijetí takové zprávy vykonáním určité činnosti (podle typu objektu). Jelikož je tato zpráva hojně využívána, je jí vyhrazen vlastní GUI prvek v podobě tlačítka (obrázek 1.3). Ten vysílá zprávu „`bang`“ po kliknutí myší nebo při příchodu jakékoliv zprávy na jeho vstup. Vizuálně je jeho aktivace doprovázena bliknutím tlačítka.

Obr. 1.3: GUI prvek pro „`bang`“ zprávu

Pdobným GUI prvkem je prvek **toggle**. Jedná se o zaškrtávací políčko, jehož výstup vyšle při změně stavu zprávu s číslem 1 nebo 0. Dalším GUI prvkem je prvek **radio**, který má celočíselný výstup v mezích $0-(N-1)$, kde N je volitelný rozsah prvku. Obdobným prvkem je **slider**, kterým lze volit čísla ze „spojitého“ rozsahu s nastavitelnou horní a dolní mezí.



Obr. 1.4: GUI prvky toggle, radio a slider

1.1.3 Objekty send, select a trigger

Pro objasnění vnitřní struktury zhotoveného nástroje je nejprve důležité stručně popsat fungování některých základních objektů. Objekt `send` spolu se svým sdruženým objektem `recieve` (lze zapsat pouze zkratkami „`s`“ a „`r`“) pracují jako neviditelné datové propojení, tzn. vstup objektu `send` je vyslán na výstup objektu `recieve` se shodným argumentem. Pro zpřehlednění výkladu nebudou tyto objekty vždy explicitně vyjmenovány, zmíněno bude pouze jimi zprostředkované propojení.

Objekt `select` má o jeden víc výstupů než argumentů a pouze jeden vstup. Zprávy na svém vstupu porovnává se svými argumenty. Pokud se zpráva na vsutpu shoduje s jedním z argumentů, z příslušného výstupu je odeslána zpráva „`bang`“. Pokud se neshoduje, je zkopírována na poslední výstup. Objekt select s argumenty `1`, `2` a `3` má tedy čtyři výstupy a v případě přijetí zprávy „`2`“ vyšle zprávu „`bang`“ na druhý výstup. Objekt `select` je možné napsat pouze zkratkou „`sel`“.

Objekt `trigger` vysílá **postupně** na výstupy zprávy podle svých argumentů, a to v sekvenci **zprava doleva**. Může jít o zprávy „`bang`“ (argument „`b`“), nebo zprávy jiných typů: číslo (argument „`f`“), symbol (argument „`s`“), seznam (argument „`l`“), ukazatel (argument „`p`“), libovolná zpráva (argument „`a`“). Zpráva ze vstupu objektu je zařazena do sekvence, pokud její typ odpovídá jednomu z argumentů. Zprávy „`bang`“ jsou z objektu vyslány při každé aktivaci. Objekt trigger je možné zapsat zkratkou „`t`“.

1.1.4 Seznamy

Zprávu sestávající z více znakových řetězců oddělených mezerami Pd interpretuje jako **seznam** (*list*). Řada objektů se seznamy pracuje, dokáže je různým způsobem dělit a jednotlivé řetězce zpracovávat separátně. Právě práce se seznamy umožňuje zhotovenému patchi číst textové soubory. Seznamy nezačínající číslem musí začínat znakovým řetězcem „`list`“, který má pouze deklarativní charakter a do datového obsahu nezasahuje. Znak čárky (`,`) v Pd znamená konec zprávy, a to i v případě seznamu. Tato skutečnost má význam pro způsob načítání obsahu textového souboru, jak bude popsáno v podkapitole 3.1.1.

1.1.5 Subpatche

Každý uložený patch ve stejném adresáři lze umístit do jiného patche v podobě objektu. Název objektu odpovídá názvu vkládaného patche. Jeho vstupy a výstupy lze vytvořit ve vkládaném patchi pomocí objektů „`inlet`“ a „`outlet`“.

Při editaci patche lze v nastavení plátna zvolit možnost „graph on parent“. V takovém případě se patch při vložení do jiného nezobrazí jako prostý objekt, ale jako plátno o určitých mezích. V patchi, do kterého je vloženo, se na plátně nezobrazí objekty, propojení, ani prvky typu zpráva. Viditelné zůstanou komentáře a GUI prvky. Velikost tohoto plátna lze nastavit.

1.2 MIDI výstup

1.2.1 MIDI protokol

Protokol MIDI (*musical instrument digital interface*) je digitální rozhraní vytvořené v roce 1983. Účelem jeho vytvoření bylo zajištění standardizovaného jazyka pro komunikaci mezi hudebními nástroji, nicméně se prokázalo jako velmi efektivní prostředek k digitální kodifikaci hudebního projevu obecně. Data jsou v protokolu přenášena *MIDI zprávami*. Tyto zprávy nepřenáší žádný audio signál, pouze předávají informace o přehrávaných tónech a interakcích s ovládacími prvky nástroje. Pro hru na hudební nástroje ovládané protokolem MIDI jsou nejdůležitější zprávy *note on* a *note off*, které značí začátky a konce přehrávaných tónů. Každá taková zpráva přenáší tři základní informace [2]:

1. číslo kanálu (MIDI protokol rozlišuje 16 indexů kanálu)
2. číslo příslušné noty v rozsahu 1–127
3. číslo odpovídající síle tónu, opět v rozsahu 1–127

Protokol MIDI se dnes hojně využívá k ovládání virtuálních nástrojů. Zprávy do programu mohou přicházet z hardwarového nástroje nebo MIDI kontroleru (zařízení jehož klávesy či ovládací prvky jsou určeny právě k vysílání MIDI zpráv). Dále mohou být v reálném čase generovány virtuálně, nebo být čteny z již nahraného a uloženého MIDI souboru.

1.2.2 MIDI v Pure Data

Nástroj vytvořený v rámci bakalářské práce generuje při čtení textového souboru MIDI zprávy v reálném čase pomocí objektu `makenote`. Objektem `noteout` je následně vysílá z Pd. V nastavení „`Media > MIDI settings...`“ lze zvolit MIDI výstup Pd. Dostupné jsou jak hardwarové výstupy připojených MIDI zařízení (např. zvukové karty), tak virtuální MIDI sběrnice. Některé softwarové nástroje disponují virtuálními MIDI vstupy, které lze také najít v tomto nastavení. Nicméně pro propojení s ostatními virtuálními nástroji nebo vstupem programu DAW (*digital audio workstation*) je v řadě případů nutné vytvořit novou virtuální sběrnici. K tomuto účelu lze využít například volně dostupné aplikace „loopMIDI“ od německého vývojáře Tobiase Erichsena [7]. V aplikaci uživatel snadno vytvoří MIDI sběrnici *(port)*, který virtuální nástroje a DAW programy rozpoznají jako vstup.

1.3 Soustava ladění a notace

Ačkoliv způsob interpretace dat z MIDI zpráv se může podle nástroje lišit, protokol MIDI byl vytvořen s úmyslem návaznosti na tónovou soustavou temperovaného

ladění.

Soustava ladění je vytvořena násobením referenčního kmitočtu základního tónu konstantami ke získání kmitočtů odpovídajících dalším tónům. Soustavy ladění se mezi sebou tedy liší poměry kmitočtů svých tónů. Temperované ladění bylo zavedeno jako korekce nežádoucích vlastností dřívějších soustav, které vycházely z racionálních poměrů kmitočtů. Příkladem takové v hudební praxi obzvlášť nežádoucí vlastnosti je fakt, že stejný tón může mít různou výšku v závislosti na volbě základního tónu soustavy. Tzn. tón a odvozený jako sexta od základního c neměl shodný kmitočet s tónem a odvozeným jako kvinta od základního d. [3]

V temperovaném ladění je každý půltón (např. klávesa piana) násobkem předchozího tónu a konstanty $\sqrt[12]{2}$. Dvanáct půltónů pak tvoří oktávu, tj. dvojnásobek původní frekvence. Poměry tónů stejných hudebních intervalů jsou si v temperovaném ladění napříč celým rozsahem a všemi stupnicemi rovny. Nejčastěji využívanou referenční frekvencí pro zavedení soustavy je 440 Hz, odpovídající tónu a'. Protokol MIDI v obvyklé praxi přiřazuje indexy z rozsahu $1-127$ tónům temperovaného ladění.

V anglickém jazykovém prostředí je využíván systém notace těchto tónů, který se od středoevropského systému notace mírně odlišuje. První odlišností je rozdílné značení výšky oktávy. Druhou odlišností je několik rozdílů v názvech tónů. Jelikož je vytvořený nástroj určen k napojení na virtuální nástroje přijímající MIDI zprávy, jsou názvy tónů vyznačeny anglickou normou. Drtivá většina virtuálních nástrojů této normy využívá. Navíc je úspornější, co se použitých symbolů týče. Rozdíl oproti středoevropské normě je vidět v přiložené tabulce A.1. Výšky oktáv nejsou ve zhotoveném softwaru značeny symboly, pouze polohami ovládacích prvků.

1.4 Sekvencery

Sekvencer je typ řídícího obvodu nástroje, ve kterém jsou zaznamenány řídící signály po pevně daných časových krocích. [4] Kroků je obvykle 8 nebo 16, v návaznosti na nejčastěji využíváné takty. Po spuštění jsou řídící informace po krocích přehrávány nastavenou rychlostí ve smyčce. Nástroj zhotovený v rámci této bakalářské práce využívá sekvenceru, který vysílá do vstupní jednotky signály pobízející k načítání znaku. Oproti běžným sekvencerům nabízí uživateli možnost nastavit délku kroku.

1.5 Kódování textu

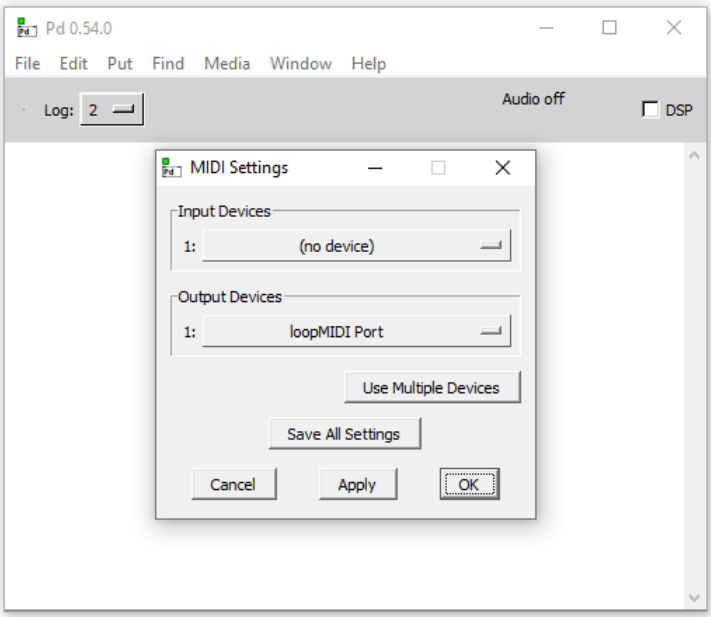
Text je v digitálním prostředí nejčastěji kódován po jednotlivých symbolech, každému znaku tak přísluší jedinečný řetězec bitů. Ve vývoji kódů určených k reprezen-

taci textu byla významnou standardizací norma **ASCII** (*American Standard Code for Information Interchange*). ASCII kóduje každý znak jako posloupnost sedmi bitů (s předřazeným bitem 0 jako adaptací pro osmibitové prostředí), čímž je schopna vyjádřit 128 unikátních symbolů. Tyto znaky a jejich kódy jsou často znázorněny pomocí ASCII tabulky. Jedná se o velká a malá písmena anglické abecedy, interpunkční znaménka, číslice a některé řídící informace. [5]

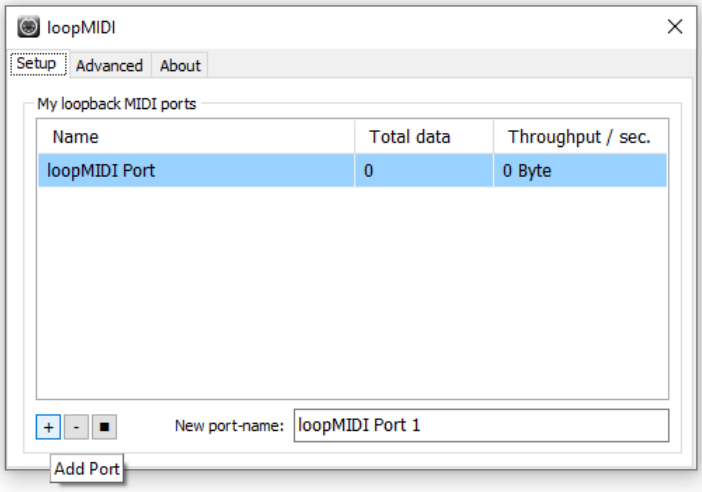
ASCII kódování ani jeho rozšíření použitím předřazeného bitu však nestačí k obsažení znaků dalších světových abeced. Za účelem jejich zakódování tak vznikl šestnáctibitový standard **Unicode**, který je schopný vyjádřit $2^{16} = 65\,536$ unikátních znaků. ASCII znakům je v něm ponechán shodný kód přenesený na šestnáct bitů. Nevýhodou Unicode je však dvojnásobná velikost dat zakódované textové informace.

Tuto nevýhodu odstraňuje kód **UTF-8**, vycházející z Unicode. Tento kód využívá jinak nezvyklou proměnnou délku kódu, kde jednotlivé znaky mohou zabírat 1, 2, nebo 3 bajty. [6] Nejhojněji využívané znaky anglické abecedy jsou tak kódovány svým osmibitovým ASCII kódem, zatímco znaky například české abecedy v něm mají svůj šestnáctibitový kód Unicode.

Pd objekt `list tosymbol` převádí řetězec znaků na seznam čísel desítkové soustavy. Výsledný kód však nevychází vždy z totožné normy. Pro znaky obsažené v ASCII tabulce je na výstupu `list tosymbol` číslo odpovídající jeho sedmibitovému ASCII kódu. Pro znaky české abecedy jsou na výstupu čísla dvě, odpovídající dvěma osmibitovým informacím – celkově tedy šestnáctibitové informaci – kódu Unicode. Objekt `list tosymbol` prokazatelně nemá žádnou metodu řešení pro všechny UTF-8 znaky, jako například „€“. Dokumentace k objektům Pd se ke kódování nevyjadřuje exaktně, pouze uvádí, že informace na výstupu „může být ASCII nebo může být Unicode“. Nicméně pro účely této bakalářské práce stačí s jistotou znát chování objektu při dekódování anglické (jedno číslo, ASCII kód) a české abecedy (dvě čísla, Unicode).



Obr. 1.5: Nastavení MIDI výstupu Pd



Obr. 1.6: Vytvoření sběrnice v aplikaci loopMIDI

2 Návrh a využití nástroje

2.1 Postup návrhu

Zpracování textových řetězců bylo zamýšleno již od začátku návrhu. Mezi prvními testovanými textovými soubory byly soubory s příponou `.pgn`, zaznamenávající průběh šachové partie. Pure Data obsahuje objekt `textfile`, určený k načítání řídících informací z textových souborů. Dalším klíčovým objektem je `list fromsymbol`, který konvertuje textové znaky na číselné kódy. Pd bylo tedy zřejmou volbou softwaru jakožto programovací prostředí určené k tvorbě hudebních nástrojů, disponující zároveň objekty vhodnými k dekódování textu.

Zvolen byl také výstup v podobě MIDI zpráv. Nástroj je unikátní především svým způsobem generace tónů ze vstupní informace. Díky MIDI výstupu je možné tuto metodu generace využít k ovládání libovolného virtuálního či hardwarového nástroje s MIDI vstupem. To programu zajišťuje širší škálu využití než kdyby nástroj generoval vlastní audio signál. Patchů syntezátorů zvuku vytvořených v Pd existuje velké množství. Jejich případné napojení na zhotovený nástroj není obtížné, obzvlášť v případech, kdy obsahují vlastní MIDI vstup.

Důležitým krokem byla volba způsobu transformace kódů textu na výstupní sekvenci. Po zavedení metronomického načítání jednotlivých znaků nejprve odpovídal sedmibitový kód ASCII přímo indexu tónu MIDI (v obou případech je rozsah 0–127). Z hudebního hlediska však taková sekvence tónů byla vždy chromatická. Obvyklým způsobem převodu na sekvenci zapadající do konkrétní tóniny je zavedení „filtru“, který tóny mimo zvolenou tóninu buď tlumí, nebo je převádí na tóny zvolené tóniny. V případě takového řešení by výšky tónů odpovídající znakům byly závislé na nastavené tónině, stejně však pevně dané. Namísto toho byl zvolen jiný postup. Ve výsledném nástroji uživatel může ovládacími prvky nastavit výšku tónu odpovídající každému jednotlivému znaku zvlášť. Který znak přehraje který tón je tak volně nastavitelné.

Úskalím tothoto přístupu je, že čím více je podporovaných znaků, tím je pro uživatele časově náročnější všem znakům přiřadit tón. Z toho důvodu byly implementovány globální ovládací prvky, kterými je možné měnit nastavení zvoleného tónu pro všechny znaky současně. Jedná se o transpozici o půltón nebo oktávu, zatlumení/odtlumení všech znaků a zatlumení znaků náhodně. Dále byla nutná implementace prvků, které by umožnily rychlé nastavení znaků na tóny zvolené tóniny. Namísto výběru z předpřipravených naprogramovaných tónin byly navrženy ovládací prvky, které umožňují náhodné nastavení tónů odpovídajících znakům. Uživatel přitom vybere množinu tónů, ze kterých je vybíráno. Pro naladění tóniny např. *c dur* tak uživatel na ovládacích prvcích zvolí její tóny a klikne na ovládací prvek, který z

nich náhdoně vybere. Náhodný výběr je proveden pro každý znak zvlášť.

Dalším krokem bylo převedení metronomického načítání znaků s nastavitelnou rychlostí na krokový sekvencer s nastavitelnou délkou jednotlivých kroků. Informace o vypočtené délce kroku byla přivedena na výstup, kde je využita k nastavení délky přehrávaného tónu. Noty na výstupu jsou tak synchronní svými začátky a konci s kroky sekvenceru.

Důležitými úkoly byly také implementace smyčky k načtení všech „zpráv“ v textovém souboru a naprogramování podpory českých znaků, které jsou kódovány dvěma čísly. Tato řešení jsou spolu s celou vnitřní strukturou nástroje popsána v kapitole 3.

Nástroj byl rozdělen do jednotlivých patchů, díky čemuž je dosaženo zvýšené přehlednosti ovládání. Nástroj se spouští a ovládá v hlavním patchi „`TEXT_Reader`“. Ten umožňuje obsluhu tří hlavních částí: vstupu, sekvenceru a znakového převodníku.

2.2 Vstup

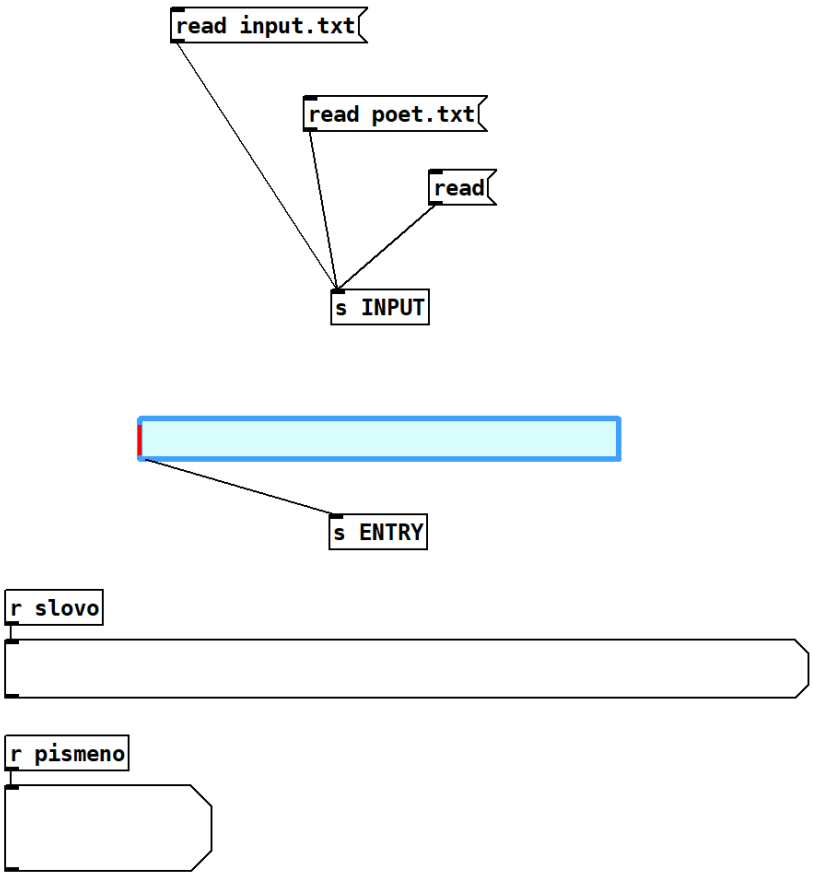
Na obrázku 2.1 je vidět vstupní část nástroje. Text může být uživatelem zadán dvěma způsoby – z textového souboru v Pd adresáři, nebo zadáním do modrého „`textentry`“ boxu.

2.2.1 Načtení souboru

K načtení souboru jsou pro uživatele nachystány prvky obsahující zprávy. Syntaxe příkazu pro načtení souboru je řetězec „`read`“ následovaný názvem souboru včetně přípony. Soubor se musí nacházet v Pd adresáři a případné změny v něm provedené musí být uloženy. V elektronické příloze jsou předpřipraveny dva vzorové soubory, které může uživatel načíst kliknutím na prvky obsahující příslušné zprávy. Pro zadání textu tak není nutné vytvářet nový textový soubor, stačí přepsat obsah např. souboru „`input.txt`“.

2.2.2 Textentry

Pd nedisponuje žádným prvkem, který by uživateli v *execute* módu dovoloval zadávat text. Box `textentry` tento nedostatek doplňuje, byť s sebou přináší rovněž určitá omezení. Nelze v něm využít klávesu *backspace* a nepodporuje specifické znaky české abecedy. Navíc při přechodu do edit módu dále zachycuje vstupy klávesnice (byť nejsou vysílány na výstup). Box `textentry` byl navržen uživateli Pd fóra a není autorskou součástí práce. [8]



Obr. 2.1: Vstupní část patche

Box `texentry` v nástroji slouží pouze pro rychlé a jednorázové (bez možnosti *loop*) zadání textového řetězce. Řetězec je vyslán na výstup boxu klávesou *enter*. Pro zpracování dlouhých řetězců textu je vhodnější metodou načtení textového souboru.

2.2.3 Indikátory slova a písmene

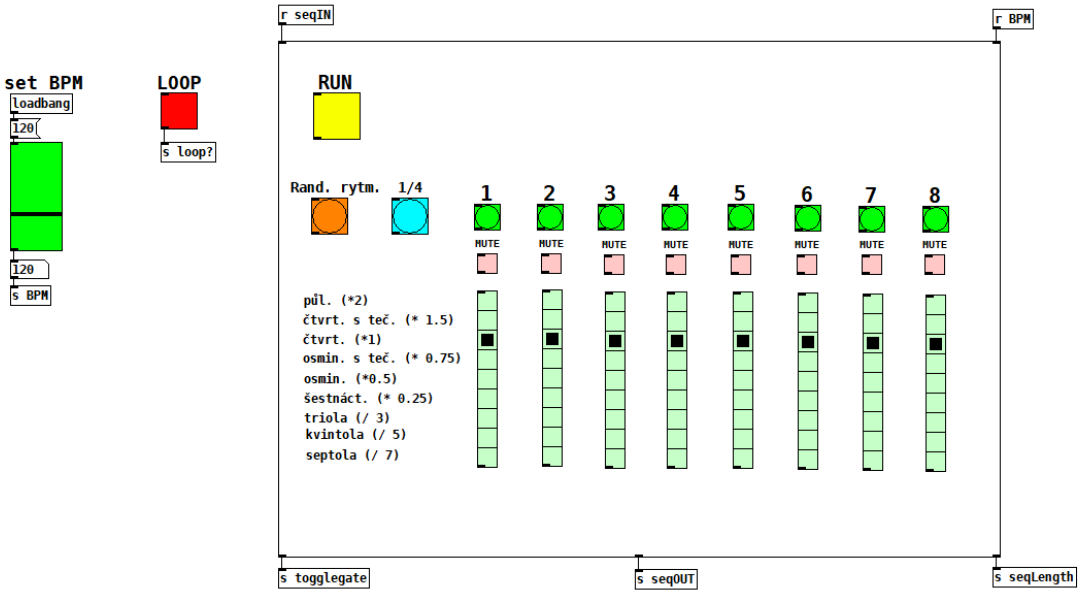
Prvky typu **symbol** zobrazují symboly a jsou považovány za GUI prvky. Prvky symbolů pod načítáním vstupu (viz obr. 2.1) slouží jako indikátory aktuálně čteného slova a aktuálně čteného znaku za chodu sekvenceru. Jednoduchou kontrolou správného načtení vstupního textu je zobrazení prvního slova ve výše položeném symbolovém prvku (to se zobrazí již před spuštěním sekvenceru).

2.3 Sekvencer

Sekvencer v patchi `TEXT_Reader` má osm kroků s nastavitelnou délkou. Ke spouštění sekvenceru slouží žlutý toggle prvek nadepsaný „RUN“. Zelené bang prvky nadepsané čísly slouží jako indikátory chodu sekvenceru. Pod nimi se nachází toggle

prvky nadepsané „MUTE“, které umožňují utlumit daný krok. Délka kroku je při tomto zatlumení zachována, v přehrané sekvenci se tak vyskytne pomlka odpovídající délky. Délka kroku lze nastavit GUI prvky typu radio. Nabízené možnosti jsou vypsány v komentáři nalevo od těchto prvků. Operátor v závorce objasňuje dobu trvání zvolené délky ve vztahu ke čtvrťové notě (viz tabulka A.2). Bang prvky nadepsané „Rand. rytm.“ a „1/4“ umožňují zvolit náhodnou délku všech kroků, respektive nastavit všechny kroky na délku čtvrťové noty.

Nalevo od ohraničeného plátna se nachází zelený prvek typu slider sloužící k nastavení B.P.M. (*beats per minute*, tedy kolik čtvrťových not sekvencer přehraje za minutu) a červený toggle prvek s možností přehrávání textu ve smyčce (pouze při čtení textového souboru).



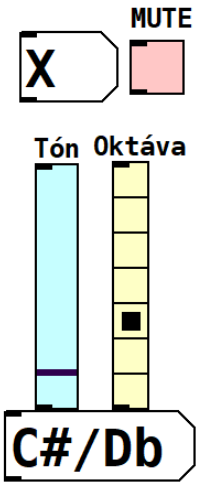
Obr. 2.2: Část patche ovládající sekvencer

2.4 Znakový převodník

Na ovládacím panelu převodníku situovaném níže v patchi `TEXT_Reader` uživatel nastavuje, jaké výšky tónu jsou přiřazeny jednotlivým znakům. Ve vrchní části panelu se nacházejí globální ovládací prvky, v části dolní pak individuální ovládací prvky ke každému podporovanému znaku.

2.4.1 Nastavení znaku

Na obrázku 2.3 jsou vidět ovládací prvky vztahující se ke znaku „X“. Znak je nadepsán v symbolovém prvku nad prvky slider a radio. Pro každý podporovaný znak lze modrým sliderem zvolit jeden z dvanácti půltónů oktávy temperovaného ladění. Žlutým radio prvkem lze pak volit výšku oktávy. Nejnižším volitelným tónem je C_1 odpovídající indexu MIDI noty 24. Nejvyšším tónem rozsahu je c'''' odpovídající indexu MIDI noty 108. Znak je možné v sekvenci utlumit zaškrtnutím červeného toggle prvku nadepsaného „MUTE“. Název zvoleného tónu se zobrazuje pod ovládacími prvky.



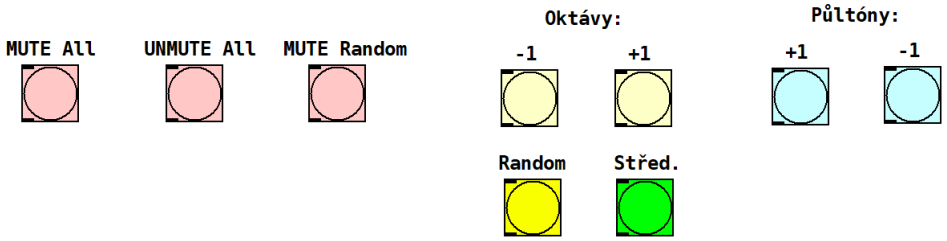
Obr. 2.3: Ovládací prvky pro znak „X“

2.4.2 Globální nastavení

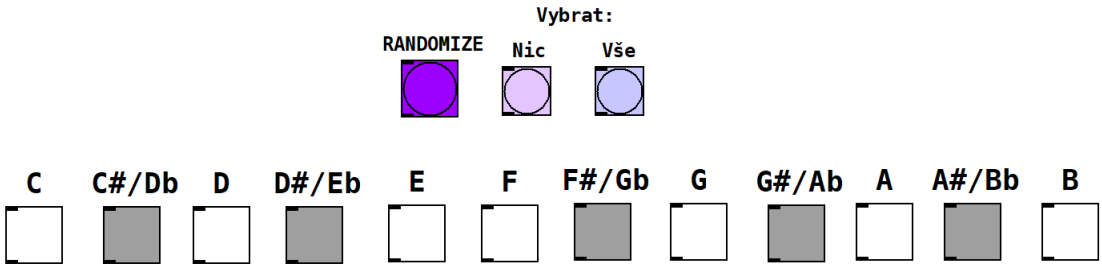
Globální ovládací prvky patche umožňují utlumit/odtlumit všechny znaky, nebo zatlumit náhodně vybrané znaky. Dále pak umožňují transpozici o jednotlivé oktávy a půtóny, zvolit oktávu pro každý znak náhodně, nebo pro každý znak nastavit oktávu na střed rozsahu. Všechny provedené změny se projeví na ovládacích prvcích jednotlivých znaků. Výjimkou je zatlumení znaku mezery, které je vždy nastaveno uživatelem a není ovládáno globálními ovládacími prvky.

Bang prvek „RANDOMIZE“ vybere tón (výška oktávy zůstane stejná) náhodně pro každý znak, přičemž vybírá z množiny tónů zvolené uživatelem. Tuto množinu uživatel vybere zaškrtnutím příslušných toggle prvků pod RANDOMIZE tlačítkem. Bang prvky „Vybrat: Nic“ a „Vybrat: Vše“ jsou nápomocné při tomto výběru, kliknutím na ně uživatel množinu vyprázdní, respektive vyplní všemi tóny. Při stisku těchto tlačítek žádný náhodný výběr neprobíhá, slouží pouze jako součást procesu

volby tónů, ze kterých posléze vybírá RANDOMIZE tlačítko. Pokud je množina tónů při stisknutí RANDOMIZE tlačítka prázdná, nejsou provedeny žádné změny.



Obr. 2.4: Globální ovládací prvky



Obr. 2.5: Náhodná volba tónů

2.5 Příklad využití

Obsahem elektronické přílohy je mimo jiné krátká skladba demonstrující konkrétní využití patche. Patch byl využit k ovládání různých virtuálních nástrojů, s různými přístupy k jeho nastavení. Na vstup byl přiveden text ze suboru obsaženého v elektronické příloze „`poet.txt`“.

Jediná část aranžmá neovládaná patchem je hi-hat smyčka. Ta tvoří v první části skladby rytmickou sekci spolu s bicím automatem. Údery bicího automatu jsou řízeny patchem. Sekvencer je nastavený na čtvrťové délky kroků. Znaky mezery jsou zatlumeny, souhlásky jsou nastaveny na tón, který spouští velký buben, samohlásky na malý buben. První polovina skladby je mimo jiné demonstrací dosažitelné rytmické synchronizace a využití patche v rámci širšího aranžmá.

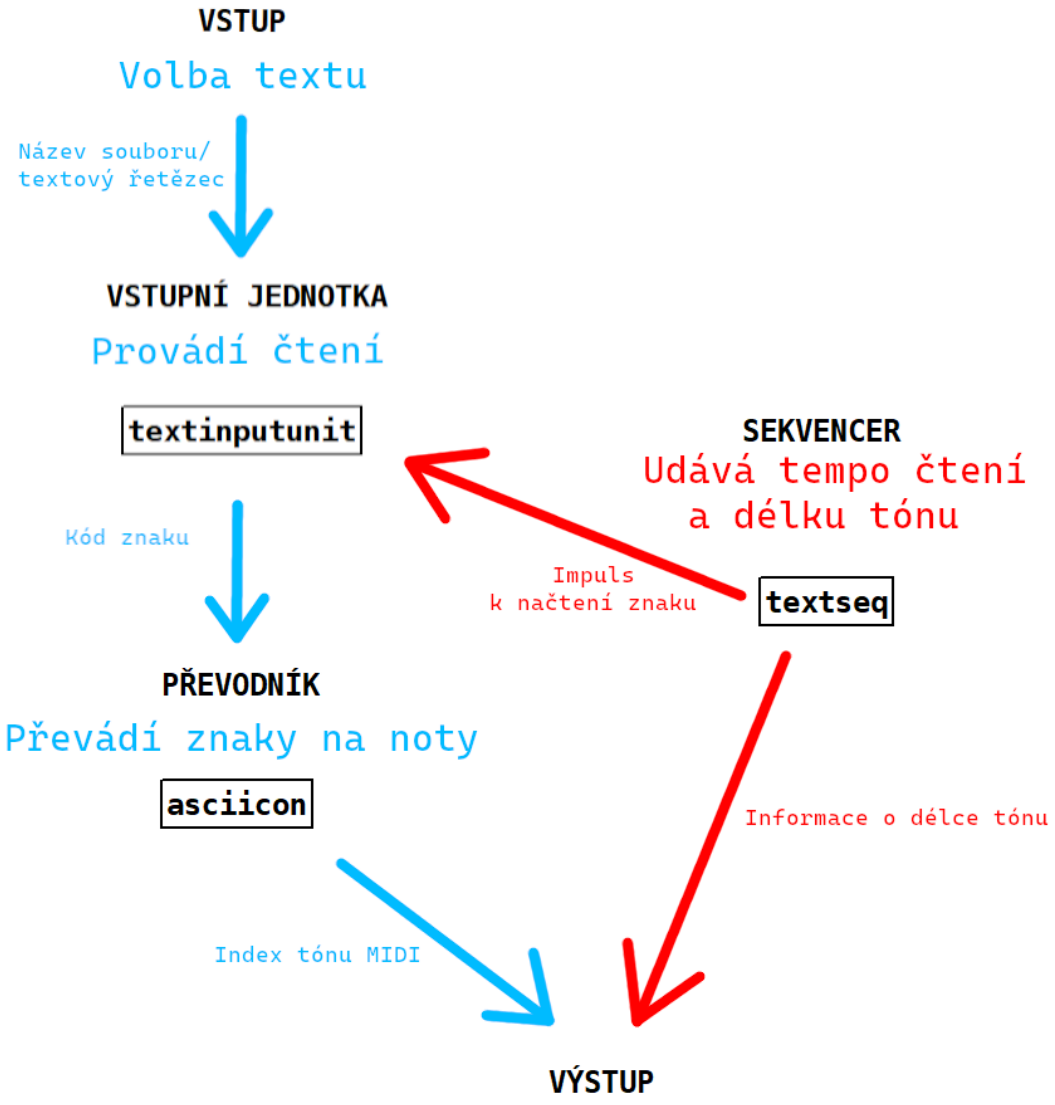
Různé syntezátory ukazují různé přístupy ke generaci tónů. Basový syntezátor je nastaven na půlové kroky sekvenceru a vybírá z malé množiny tónů. Žádné kroky sekvenceru ani znaky nejsou zatlumeny. Vzniká tak hudební informace podobná informaci generované za pomocí běžných sekvencerů. Další syntezátory se mimo své

barvy liší nastavením oktávového rozsahu patche, množstvím použitého zatlumení kroků, nastavením znaků a délkami kroku sekvenceru. V první části skladby syntezátor využívá melodičtější přístup – oktávový rozsah je úzký, více kroků či znaků je zatlumených, kroky jsou svou délkou synchronizovány se zbytkem aranžmá. Druhá polovina skladby pak ukazuje manipulaci s nastavením za chodu sekvenceru k dosažení generace zvukových ploch, bez důrazu na rytmickou synchronizaci.

Nástroj uživateli nabízí vysokou míru kontroly nad pravidly převodu jednotlivých znaků na tóny. Globální ovládací prvky jsou nápomocné k rychlým změnám nastavení celého ovládacího panelu.

3 Architektura programu

Vytvořený softwarový nástroj sestává z jednoho hlavního Pd patche a pěti subpatchů. Jak bylo popsáno výše, hlavní patch `TEXT_Reader` obsahuje GUI prvky k obsluze nástroje. Jedním ze subpatchů je již zmíněný box `textentry`. Dalšími subpatchi jsou vstupní jednotka „`textinputunit`“, sekvencer „`textseq`“, znakový převodník „`asciicon`“ a uvnitř něj se nacházející jednotka na zpracování znaku „`symbolproces`“. Na obrázku 3.1 je vidět zjednodušené grafické znázornění vztahu mezi patchi a celkové logiky fungování nástroje.



Obr. 3.1: Grafické znázornění architektury patche `TEXT_Reader`

3.1 Vstupní jednotka

Patch vstupní jednotky má tři vstupy a čtyři výstupy. První vstup slouží k přijetí příkazů ke čtení textového souboru (např. „`read input.txt`“). Druhý je napojený k výstupu boxu `textentry` a přijímá z něj znakový řetězec. Třetí vstup přijímá zprávy „`bang`“ ze sekvenceru, které pobízejí jednotku k načtení dalšího znaku. První výstup odesílá zprávu „`bang`“ značící konec čteného textu. Druhý výstup vysílá číselný kód čteného znaku. Třetí a čtvrtý výstup jsou napojeny na indikátory aktuálně čteného slova a písmene (3.1.4).

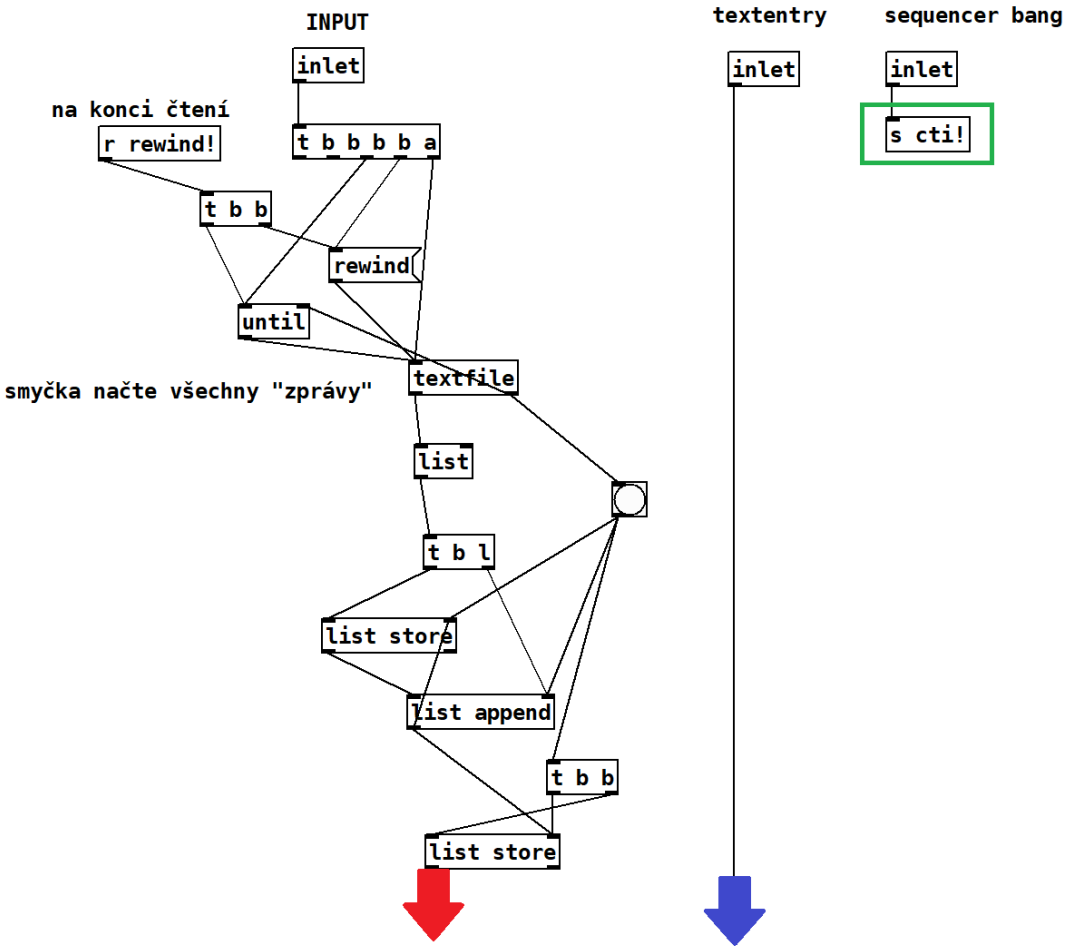
Činnost jednotky lze pomyslně rozdělit na tři hlavní části: smyčka načítající všechny „zprávy“ v souboru, část načítající jednotlivá slova a část načítající jednotlivé znaky a převádějící je na jejich číselné kódy. Vedlejší je pak poslední část, ovládající indikátory čteného slova a znaku.

3.1.1 Načtení všech „zpráv“

Na obrázku 3.2 je vidět vstupní část subpatche `textinputunit`. Příkaz „`read`“ prochází výstupem objektu `trigger` „`a`“ (*anything*) a ovládá objekt „`textfile`“, který načítá textový soubor. Po načtení je potřeba objekt přetočit na začátek souboru příkazem „`rewind`“, což zajišťuje „`bang`“ zpráva, druhá v sekvenci vycházející z `trigger` objektu. Vlastností objektu `textfile` je, že na svůj výstup po přijetí zprávy „`bang`“ vyšle další „zprávu“ v textovém souboru. Načítaný text je totiž chápán jako sekvence zpráv oddělených čárkami (jak již bylo zmíněno v první kapitole práce, čárka v Pd ukončuje zprávu). Z tohoto důvodu je po načtení a přetočení textu třetí „`bang`“ zprávou spuštěna smyška `until`, která vysílá „`bang`“ zprávy na vstup objektu `textfile` dokud není zastavena zprávou „`bang`“ z jeho pravého výstupu, která značí konec čtení souboru.

„Zpráva“ je napřed uložena v objektu `list append`, přivedením na jeho pravý vstup. Poté je vyslána zpráva „`bang`“ do objektu `list store`, který v sobě má uložen prázdný seznam a pošle jej na svůj výstup. Objekt `list append` v sobě uložený seznam přidává k seznamu přivedenému na svůj levý vstup. Výsledný seznam je uložen v dalším objektu `list store` a zároveň v objektu `list append`. Při příchodu další zprávy je k ní tedy objektem `list append` přidán dosavadní seznam. Na konci čtení „zpráv“ z textového souboru vyšle objekt `textfile` z pravého výstupu zprávu „`bang`“. Ta zastaví `until` smyčku, vymaže objekty `list store` a `list append` využívané při iteraci a aktivuje `trigger` objekt, který nejprve vyšle celý uložený seznam v druhém `list store` objektu do další části patche a poté jej rovněž vynuluje „`bang`“ zprávou vyslanou do jeho pravého vstupu. Výsledkem této iterace je celý text uložený jako seznam slov, znaky čárek se v něm však již nevyskytují.

Jejich absenci je věnována část poslední kapitoly práce.

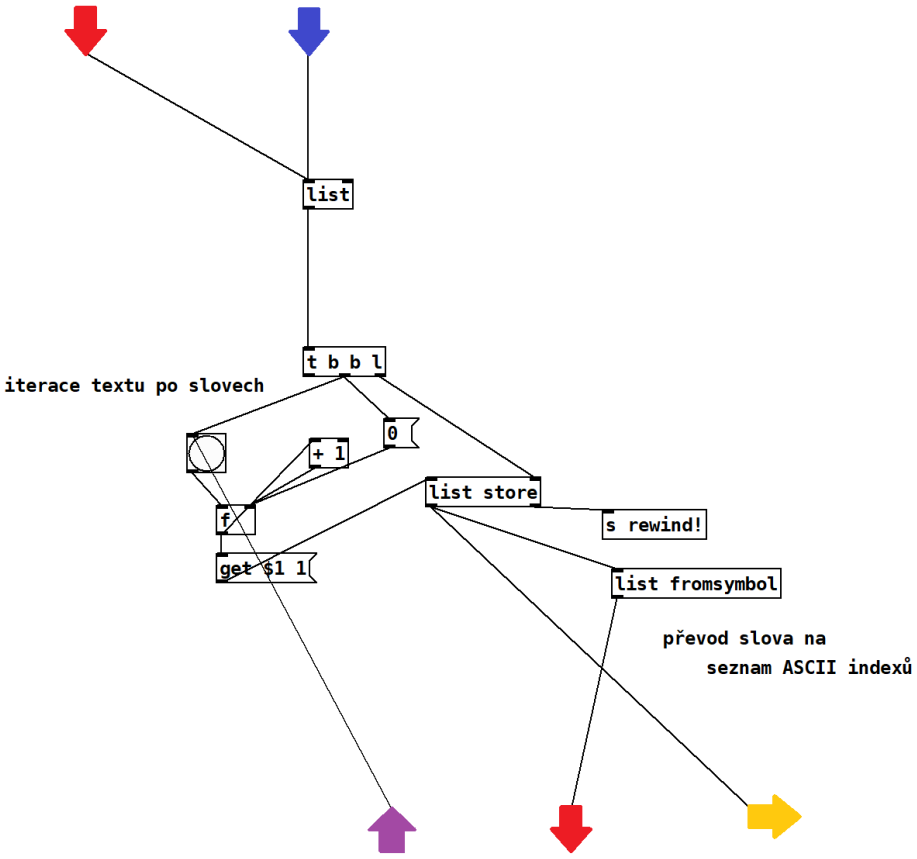


Obr. 3.2: První část patche `textinputunit`

3.1.2 Iterátor ke čtení slov

Na obrázku 3.3 je vidět další část vstupní jednotky. Seznam je nejprve uložen v objektu `list store`. Následně je vynulován objekt `f` (*float*), obsahující číselný iterátor. Každá „`bang`“ zpráva přivedená na objekt s uloženým iterátorem jej pomocí objektu `+ 1` navýší o 1. První z těchto „`bang`“ zpráv vychází z objektu `trigger`, další přicházejí z následující části patche. Iterátor zvyšuje první argument zprávy „`get`“, kterou jsou z objektu `list store` postupně získávány prvky seznamu – slova. Slovo je z výstupu `list store` vysláno do objektu `list fromsymbol`, který jej převede na seznam čísel odpovídajících kódům jeho znaků. Na konci textu se na pravém výstupu `list store` objeví zpráva „`bang`“. Ta je vyslána na začátek vstupní jednotky kde přetočí objekt `textfile` a také na první výstup patche.

V případě použití boxu textentry je zadaný text vyslán přímo do této části vstupní jednotky (modrá šipka v obrázcích 3.2 a 3.3).

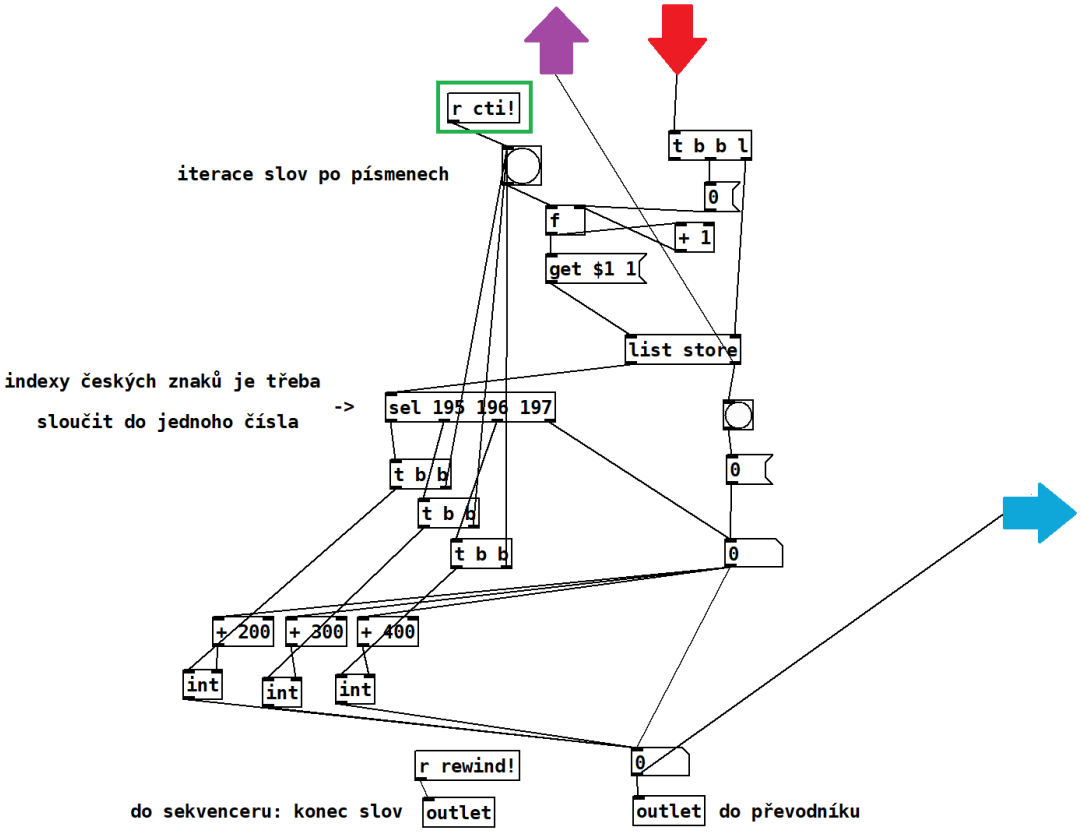


Obr. 3.3: Iterace textu po slovech

3.1.3 Iterátor ke čtení znaků

Impulsy sekvenceru jsou přijímány v následující části vstupní jednotky, která je vidět na obrázku 3.4. `Send` a `recieve` objekty s argumentem „`cti!`“ spojují tuto část s třetím vstupem patche. Obdobným způsobem jako v předcházející části jsou zprávou „`get`“ se zvyšujícím se argumentem získávány položky seznamu kódů znaků uloženého v objektu `list store`. Zpráva je spuštěna s každou zprávou „`bang`“ přicházející ze sekvenceru. Na konci seznamu kódů (aktuálního slova) vyšle `list store` zprávu „`bang`“ ze svého pravého výstupu, která je vyslána do předchozí části jako pobídka k načtení dalšího slova (spojení vyznačeno fialovou šipkou). Tato zpráva také vynuluje číslo na druhém výstupu patche. Krok sekvenceru, odpovídající mezeře v textu, tak vysílá na druhý výstup vstupní jednotky zprávu s číslem 0. Tento krok je jednou proveden i na samém konci čtení textu.

Znaky specifické pro českou abecedu (*ě, š, č, ř, ž, ý, á, í, é, ú, ů, ď, ť, ň, ó*) jsou reprezentovány dvěma čísly podle svého Unicode kódování, tedy dvěma položkami seznamu. To pro chod programu není žádoucí. Čtené položky seznamu jsou proto filtrovány objektem `select`. První číslo, odpovídající prvním osmi bitům kódování, je pro všechny české znaky jedo z čísel 195, 196, nebo 197. V případě, že je načteno jedno z těchto čísel, je namísto vyslání kódu na výstup okamžitě vyslána `trigger` objektem „`bang`“ zpráva na vstup objektu `f`, čímž je načten následující znak. K následujícímu znaku je pak přičtena hodnota 200, 300, nebo 400 podle znaku předcházejícího. Například český znak „á“, kódovaný čísly 195 a 129 je tak na výstupu patche reprezentován číslem 329 ($195 \rightarrow 200$; $129 \rightarrow 129$; $200 + 129 = 329$). Tímto způsobem je každý znak kódován jedním číslem a žádné kódy znaků se nepřekrývají.

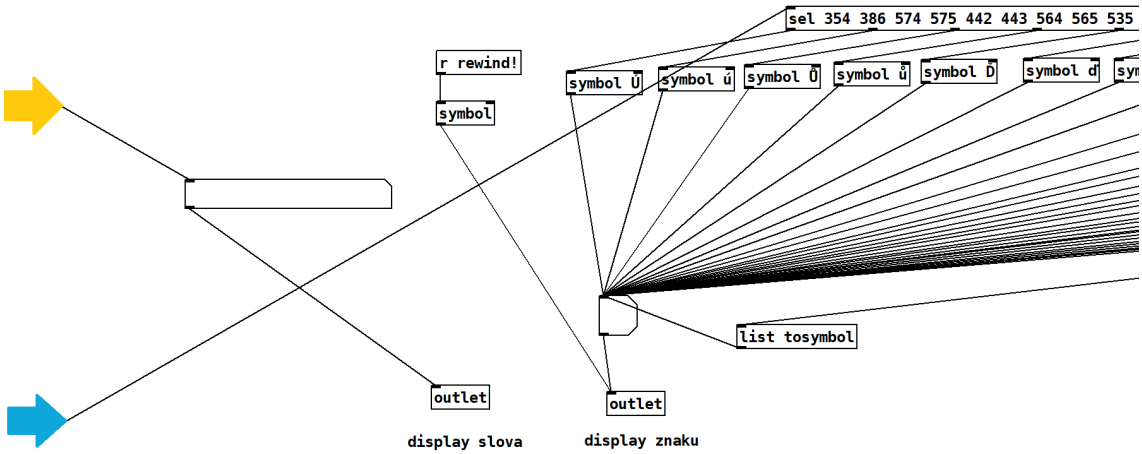


Obr. 3.4: Iterace slova po písmenech

3.1.4 Indikátory slova a znaku

Na obrázku 3.5 jsou vidět třetí a čtvrtý výstup patche `textinputunit`. Pro určení aktuálně čteného zanku je proveden zpětný převod z kódu na symbol objektem `list`

`tosymbol`. To však nemůže platit pro české znaky, které jsou v rámci patche kódovány nově zavedeným způsobem popsaným v 3.1.3. Proto je číslo znaku filtrováno objektem `select` s argumenty odpovídající novým kódům. I do této části patche je vysílána zpráva značící konec textu, aby mohl být na konci čtení vyprázdněn indikátor právě čteného znaku.



Obr. 3.5: Výstupy sloužící indikátorům

3.2 Sekvencer

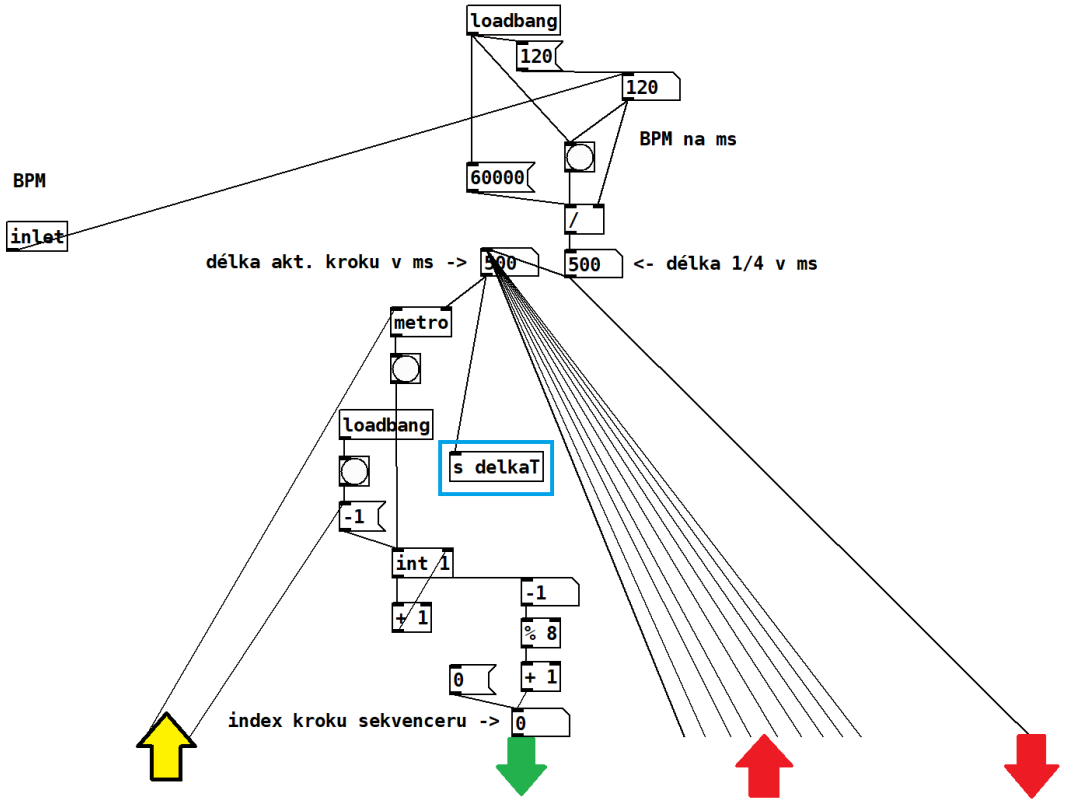
Subpatch `textseq` má dva vstupy. První je vyhrazen zprávě „`bang`“, kterou vyšle vstupní jednotka na konci čteného textu. Tato zpráva vede ke žlutému toggle prvku „RUN“ a zastavuje tak chod sekvenceru. Průchodu zprávy lze zamezit zaškrtnutím toggle prvku „LOOP“ jehož hodnota je invertována a řídí logickou bránu `spigot`. V případě zamezení průchodu této ukončovací zprávy na konci souboru začne vstupní jednotka číst text od začátku. Druhým vstupem sekvencer přijímá hodnotu B. P. M. nastavenou uživatelem pomocí zeleného slider prvku.

První výstup sekvenceru vysílá stav ovládacího prvku „RUN“. Ten je využit na výstupu celého nástroje k utlumení, pokud není sekvencer spuštěn. Druhý výstup vysílá do vstupní jednotky zprávu „`bang`“ a pobízí ji tak k načtení kódu znaku. Třetí výstup vysílá k výstupu informaci o délce aktuálního kroku v ms.

3.2.1 Jádro sekvenceru

Objekt `metro` posílá na svůj výstup periodicky zprávu „`bang`“. Je spouštěn a vypínán zprávou „`bang`“ do svého levého vstupu, ta přichází z ovládacího toggle prvku

„RUN“ Interval mezi výstupy v ms je udáván na jeho pravém vstupu. Na obrázku 3.6 je vidět jádro sekvenceru, vybudované kolem objektu `metro`. V horní části je hodnota B. P. M. převedena na délku čtvrťové noty v ms. Počet ms v jedné minutě je 60000, B. P. M. značí počet čtvrťových not za minutu, tedy 60000 vydělených hodnotu B. P. M. je počet ms v jedné čtvrťové notě. Prvek čísla s touto hodnotou délky komunikuje se součástí patche `textseq`, která slouží k výpočtu délky aktuálního kroku, ta je uložena v dalším objektu typu číslo. Tato délka je vysílána na třetí výstup patche a k pravému vstupu objektu `metro`, čímž je synchronizována délka MIDI zprávy na výstupu nástroje a délka kroku sekvenceru (čas mezi „`bang`“ zprávami `metro` objektu). Zprávy „`bang`“ na výstupu objektu `metro` zvyšují po jedné celočíselný iterátor uložený v objektu `int`. Operací modulo („%“) je tento iterátor převeden na index kroku sekvenceru.

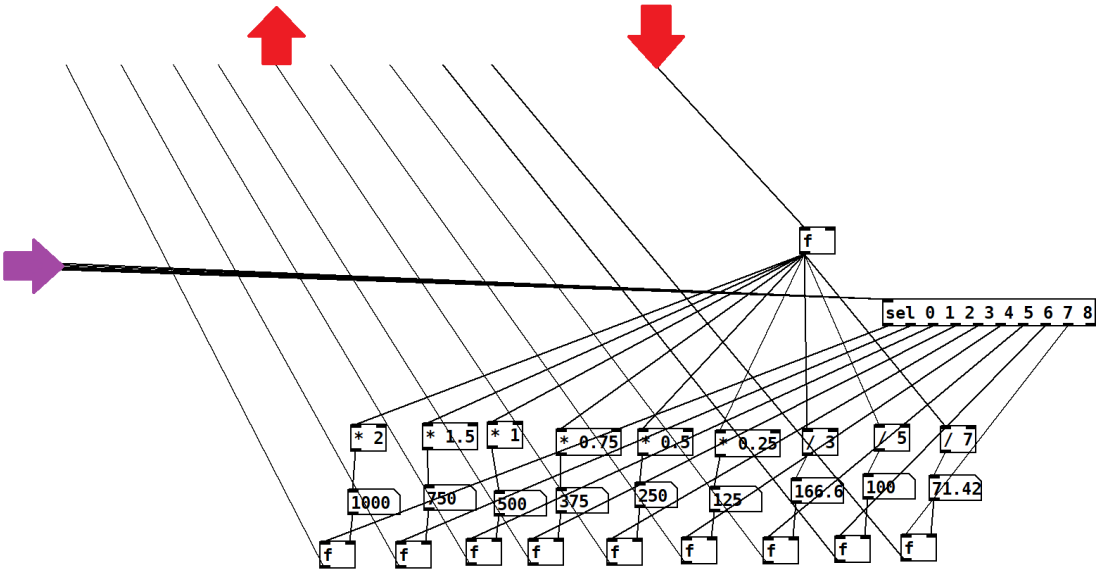


Obr. 3.6: Jádro sekvenceru

Část sloužící k výpočtu délky aktuálního kroku je vidět na obrázku 3.7. Všechny uživatelem volitelné hodnoty délky kroku jsou vypočteny v momentě změny B. P. M. a uloženy v objektech `f`.

Jejich vyslání k pravému vstupu objektu `metro` je řízeno objektem `select`, s argumenty odpovídající jednotlivým hodnotám ovládacích prvků k volbě délky kroku. Za chodu sekvenceru každý jeho krok vyšle hodnotu svého ovládacího prvku délky

kroku na vstup `select` objektu. Ten vyšle „`bang`“ na levý vstup příslušného objektu `f` a tím je délka kroku nastavena.

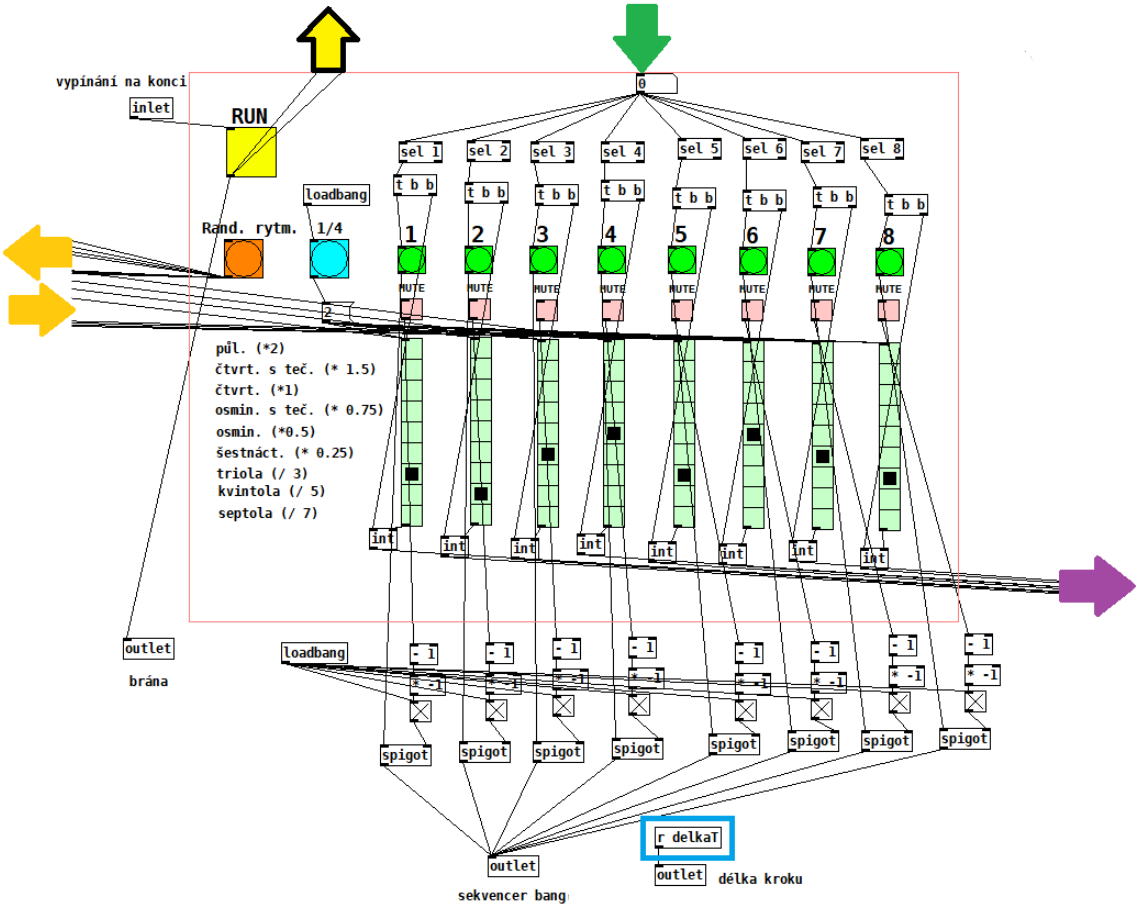


Obr. 3.7: Výpočet délky kroku

3.2.2 Ovládací panel sekvenceru

Ovládací panel a výstupy subpatche `textseq` jsou vyobrazeny na obrázku 3.8. Index aktuálního kroku je vybrán odpovídajícím prvkem `select`. Následně je objektem `trigger` vyslána sekvence dvou „`bang`“ zpráv. Napřed je získána uživatelem nastavená hodnota v ovládacím prvku radio, uložená v objektu `int` pod ním, a vyslána k části vypočítávající délku kroku v ms. Poté je zpráva „`bang`“ přes logickou bránu `spigot` vyslána na druhý výstup patche a pobízí vstupní jednotku k načtení kódu znaku. Tato brána je přes invertující operaci napojena na ovládací prvek toggle „MUTE“ příslušného kroku.

Ovládací prvek nadepsaný „1/4“ je napojen k prvku zprávy, která pro všechny radio prvky nastaví jejich třetí polohu, odpovídající čtvrťové notě. Ovládací prvkek „Rand. rytm.“ komunikuje s osmi objekty `random`, které vybírají polohu radio prvků náhodně (komunikace je znázorněna oranžovými šipkami). Změny prováděné na ovládacích prvcích délky kroku jsou přivedeny na pravý vstup objektů `int` pod nimi, aby bylo zamezeno jejich vysílání v okamžiku změny.



Obr. 3.8: Ovládací panel a výstupy sekvenceru

3.3 Převodník

Subpatch převodníku `asciicon` má pouze jeden vstup a jeden výstup. Na jeho vstup přichází ze vstupní jednotky kód znaku. Na jeho výstup je vysílán MIDI index tónu, který znaku podle nastavení odpovídá. V patchi jsou rozmístěny globální ovládací prvky a ovládací prvky jednotlivých znaků. Informace jak z globálních tak z konkrétních ovládacích prvků přijímají subpatche `symbolproces`. Pro každý znak existuje vlastní iterace tohoto subpatche a každý tento subpatch je spojený s ovládacími prvky svého znaku. Pokyny z globálních prvků jsou vysílány objekty `send` a přijímají je všechny tyto subpatche zároveň.

Převod probíhá ve dvou krocích. Před každým subpatchem `symbolproces` je předřazen objekt `select` s argumentem rovným kódu znaku (u písmen dva argumenty pro velký a malý znak). Tento `select` objekt je napojený na vstup patche, a pokud kód odpovídá, vyšle do prvního vstupu subpatche `symbolproces` zprávu „`bang`“. Vyslání zprávy „`bang`“ do subpatche `symbolproces` náležejícího zakódovanému znaku je prvním krokem převodu. Uvnitř subpatche `symbolproces` je vypo-

čítáván a uložen index MIDI tónu, který znaku podle nastavení odpovídá. Druhým krokem převodu je vyslání této uložené hodnoty na výstup subpatche `symbolproces`.

Jako první bude rozebrána vnitřní struktura patche `symbolproces`, zahrnující proces výpočtu MIDI indexu a komunikaci s ovládacími prvky převodníku.

3.3.1 Procesor symbolu

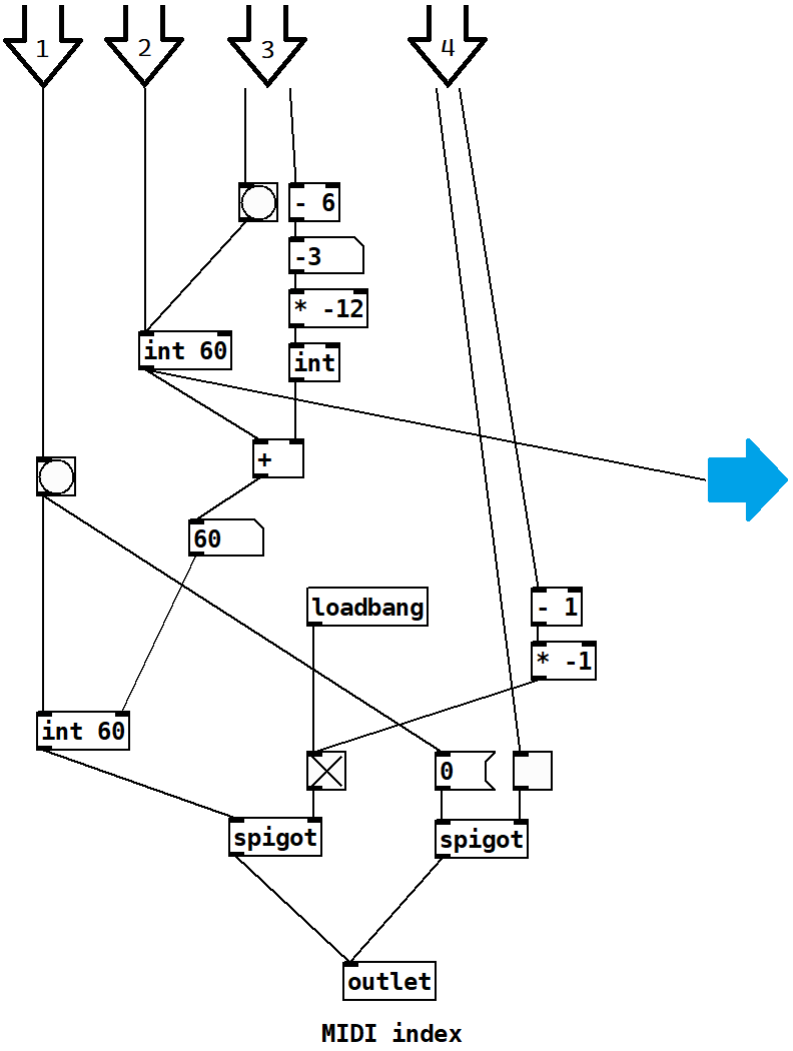
Každý patch `symbolproces` má deset vstupů a čtyři výstupy. Jeho vstupy jsou následující:

1. „`bang`“ v případě čtení znaku
2. ovládací prvek výběru tónu
3. ovládací prvek výběru oktávy
4. ovládací prvek „MUTE“ znaku
5. globální ovládací prvek ke snížení oktávy
6. globální ovládací prvek ke zvýšení oktávy
7. globální ovládací prvek ke snížení tónu
8. globální ovládací prvek ke zvýšení tónu
9. seznam sestávající z množiny indexů MIDI tónů zvolených k náhodnému výběru
10. příkaz k nastavení náhodné/střední oktávy

Jak již bylo zmíněno, první vstup je přes objekt `select` napojen na vstup patche `asciicon` a tudíž na druhý výstup vstupní jednotky `textinputunit`. Tento vstup přijímá zprávu „`bang`“, v případě, že je čten odpovídající symbol. Dalších sedm vstupů je napojeno na ovládací prvky. Devátý vstup přijímá seznam tónů určených k náhodnému výběru. Tento seznam je zkompletován mimo patch `symbolproces` a je tedy na vstupu všech procesorů symbolu stejný. Samotný náhodný výběr ze seznamu je ale proveden separátně v každém subpatchi `symbolproces`, čímž je zajištěno, že je pro každý znak vygenerována nezávislá náhodná hodnota. Poslední vstup je napojen na ovládací prvky k nastavení náhodné/střední oktávy. Tyto prvky vysílají číslo 1 nebo 2, které od sebe příkazy odlišuje.

První výstup subpatche slouží k vysílání MIDI indexu tónu po provedeném převodu. Další dva výstupy jsou navedeny zpět na vstup ovládacích prvků tónu a oktávy příslušného znaku. Čtvrtý výstup vysílá ze subpatche symbol zvolené noty.

První část subpatche `symbolproces` je na obrázku 3.9. Zabývá se **výpočtem indexu MIDI z ovládacích slider prvků** oktávy a tónu daného znaku. Je napojena na první čtyři vstupy subpatche a na jeho první výstup. Zpráva „`bang`“ z prvního vstupu subpatche vyšle buď uložený vypočtený MIDI index z objektu `int`, nebo zprávu s číslem 0. Která z těchto cest je otevřena je určeno dvěma vůči sobě inverzně zapojenými logickými branami `spigot`, ovládanými informací ze čtvrtého



Obr. 3.9: Výpočet MIDI indexu pro výstup

vstupu patche („MUTE“). Hodnota ovládacího prvku tónu z druhého vstupu je uložena v dalším objektu `int`. Z hodnoty ovládacího prvku oktávy na třetím vstupu, který má rozsah 0–6, je vypočten počet půltónů k přičtení. Hodnota je objekty s matematickými operátory otočena, aby nejvyšší hodnota vizuálně odpovídala nejvyšší poloze ovládacího prvku (přirozeně je tomu naopak). Počet oktáv násobený dvanácti se sečte s hodnotou ovládacího prvku tónu při každé změně na jednom z ovládacích prvků oktávy nebo tónu. Vypočtené číslo nastavuje hodnotu uloženou v objektu `int` v cestě „`bang`“ zprávy z prvního vstupu. Pravým vstupem objektu `int` se v něm změní uložená hodnota bez vyslání na výstup, čímž je zamezeno nechtěnému znění tónů při změně polohy ovládacích prvků.

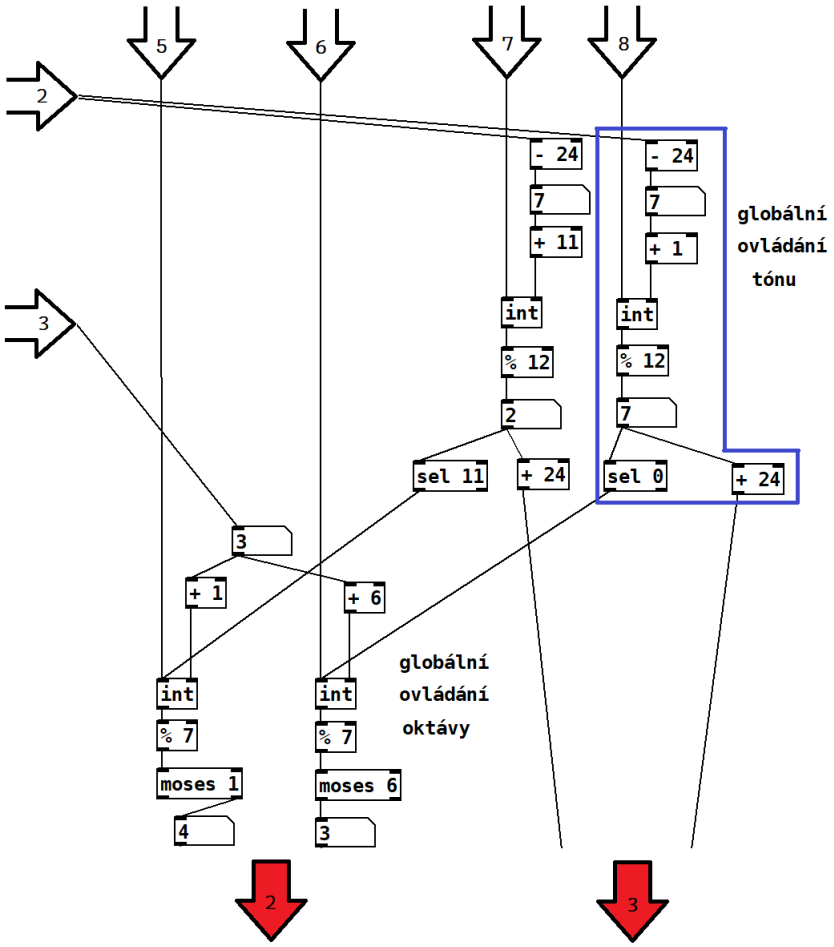
Část patche `symbolproces`, vyhodnocující **globální posouvání o jeden tón nebo jednu oktávu**, je vidět na obrázku 3.10. Princip činnosti všech čtyř větví,

připojených na jednotlivé vstupy je obdobný. Zaměřme se nejprve na navýšení tónu o jeden půltón (vyznačeno modře). Odečítání a přičítání hodnoty 24 na začátku a konci větve je zavedeno z důvodu zvoleného rozsahu ovládacího slider prvku. Díky této transpozici nabývá slider hodnot z rozsahu 0−11. Tyto hodnoty jsou „spojité“, ale objekt `int` uchovává pouze celočíselnou část. Do pravého vstupu objektu `int` je přivedena hodnota ovládacího prvku přes objekt s operátorem `+1`, objekt `int` v sobě tedy uchovává hodnotu ovládacího prvku o jedna vyšší, než je nastavená, aniž by ji vyslal na svůj výstup. Pouze v momentě příchodu zprávy „`bang`“ na levý vstup objektu z osmého vstupu patche (příkaz k navýšení tónu) je hodnota vyslána na výstup objektu. Ten je napojen na operaci `%12`. Pokud je nastavená hodnota v rozsahu $0-10$, operace modulo nemá vliv a hodnota o jeden půlton navýšená je vyslána na třetí výstup patche, spojený s ovládacím prvkem tónu. Pokud je nastavená hodnota 11 a navýšená tedy 12, na výstupu operace modulo bude číslo 0. To znamená, že při navýšení z tónu h se ovládací prvek vrátí na nejnižší hodnotu rozsahu, tzn. c. Zároveň v tomto případě objekt `select` s argumentem 0 vyšle zprávu „`bang`“ směrem k větvi navyšující oktávu.

Odečítající větev pracuje podobně. Hodnota o jedno nižší je s ohledem na operaci modulo získána přičtením 11 (díky tomu není třeba zpracovávat záporná čísla). Pokud je zvolená hodnota 0, při příchodu příkazu ke snížení tónu přijde na výstup operace modulo hodnota 11, což odpovídá skoku z tónu c na tón h. Přitom `select` objekt vyšle „`bang`“ zprávu přikazující snížení oktávy.

Přičítání a odečítání oktávy pracuje s rozsahem 0−6. Příkazy v podobě „`bang`“ zprávy nedostává jen při stisku patřičných ovládacích prvků, ale i při přechodu z jedné oktávy do druhé při navyšování/snižování tónu. Tam, kde v případě ovládání tónu dochází ke skokové změně z nejnižší polohy ovládacího prvku na nejnižší – či naopak – tomu tak u posouvání oktávy není. Na konci větví jsou umístěny objekty `moses`. Objekty `moses` pracují jako děliče čísel podle svého argumentu. Čísla vyšší nebo rovny argumentu jsou vyslány z pravého výstupu, nižší z levého. V tomto případě slouží jako omezovače. Při dosažení nejnižší oktávy tedy už ovládací prvek na příkaz k odečtení oktávy nereaguje, obdobně je tomu tak při přičítání k nejvyšší oktávě.

Na obrázku 3.11 je vidět část subpatche, která **náhodně vybírá tóny ze seznamu** při stisku globálního ovládacího prvku „RANDOMIZE“. Dále provádí **náhodnou změnu oktávy** nebo její nastavení zpět na střed. Při aktivaci prvku „RANDOMIZE“ přichází na devátý vstup patche `symbolproces` seznam, obsahující indexy MIDI tónů zvolených uživatelem. Seznam je nejprve po průchodu objektem `trigger` uložen v objektu `list store`, aniž by byl vyslán na jeho výstup (pravým vstupem obdobně jako např. u `int`). Zároveň je získáná délka tohoto seznamu objektem `list length` a nastavena jako argument objektu `random`. Poté je objektem



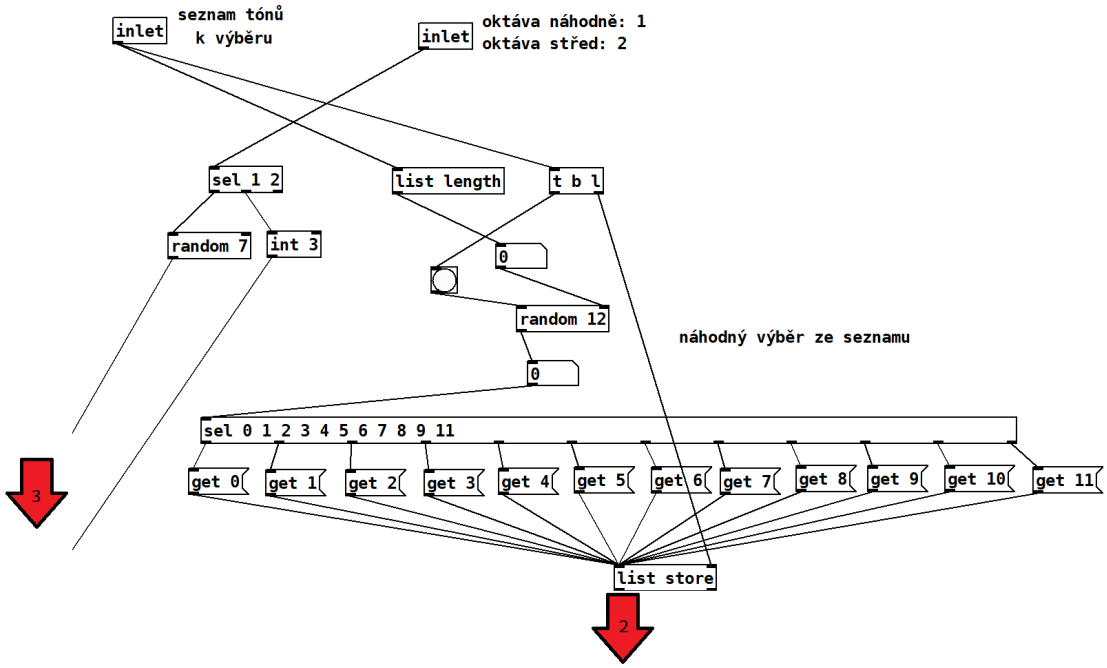
Obr. 3.10: Část procesoru znaku zabývající se globálními ovládacími pokyny

`trigger` vyslána zpráva „`bang`“ na levý vstup objektu `random`. Ten vygeneruje náhodně číslo z rozsahu 0–$(N-1)$, kde N je délka seznamu (generuje tedy N možných hodnot, jsou pouze indexovány od 0). Objekt `select` přijme tento index položky seznamu a vyšle „`bang`“ do objektu s příslušnou zprávou `get`. Ta podle svého argumentu po příchodu na vstup objektu `list store` vyšle na jeho výstup patřičnou položku seznamu. Tento index tónu je pak vyslán na druhý výstup subpatche, tedy na vstup ovládacího prvku.

Z ovládacích prvků náhodné změny oktávy nebo nastavení na střed přicházejí na desátý vstup subpatche čísla 1 nebo 2. Podle nich objekt `select` vyšle zprávu buď na objekt `random` s argumentem 7 nebo na objekt `int` s uloženou hodnotou 3. Na výstup 2 napojený na radio ovládací prvek oktávy je tak vyslána buď náhodná hodnota z rozsahu $0-7$, nebo hodnota 3 (střed rozsahu).

Jak krokové změny pomocí globálních prvků, tak náhodný výběr, provádí změny posunutím ovládacích prvků tónu nebo oktávy. První popsaná část subpatche

`symbolproces` (obrázek 3.9) zareaguje na změnu ovládacího prvku stejně, jako by ji změnil uživatel a pro první výstup je vypočtena nová hodnota.



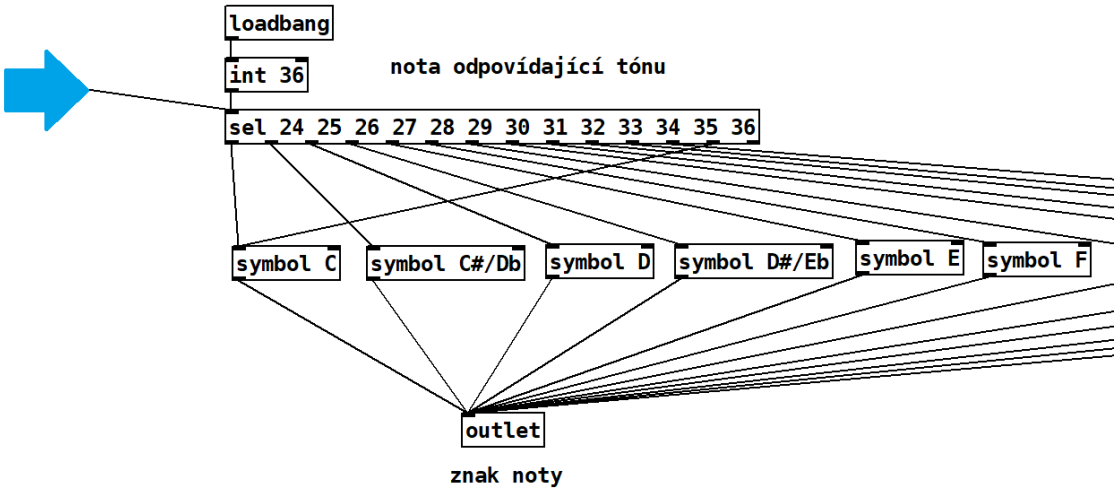
Obr. 3.11: Náhodný výběr tónů ze seznamu

Podle hodnoty ovládacího prvku tónu přijímaného druhým vstupem subpatche je **určen symbol noty** (obrázek 3.12). Hodnoty z rozsahu ovládacího prvku jsou shodné s argumenty objektu `select`. Ten vyšle při změně ovládacího prvku tónu „`bang`“ zprávu objektu s odpovídajícím symbolem noty.

3.3.2 Globální část převodníku

Jak bylo popsáno, vstup a výstup subpatche `asciicon` jsou napojeny k iteracím subpatche `symbolproces`. Uvnitř patche `asciicon` jsou pak rozmístěny všechny ovládací prvky znaků spolu s globálními ovládacími prvky. Ovládací prvky vysílají příkazové zprávy do jednotlivých procesorů znaku. Důležitou částí globální části převodníku je také sestavení seznamu MIDI indexů pro náhodný výběr.

Napojení procesorů na ovládací prvky je z důvodu úspory místa v patchi uspořádáno značně nepřehledně (obrázek 3.13). Přehledněji uspořádané propojení je na obrázku 3.14. Všechny objekty `recieve` přijímají zprávy z globálních ovládacích prvků (obrázek 3.15). Prvky „MUTE All“ a „UNMUTE All“ vysílají logické zprávy 1 a 0, které určují stav toggle prvku „MUTE“ pro každý znak. Prvek „MUTE Random“ vysílá ke všem znakovým jednotkám zprávu „`bang`“. Ta je napojena na objekt



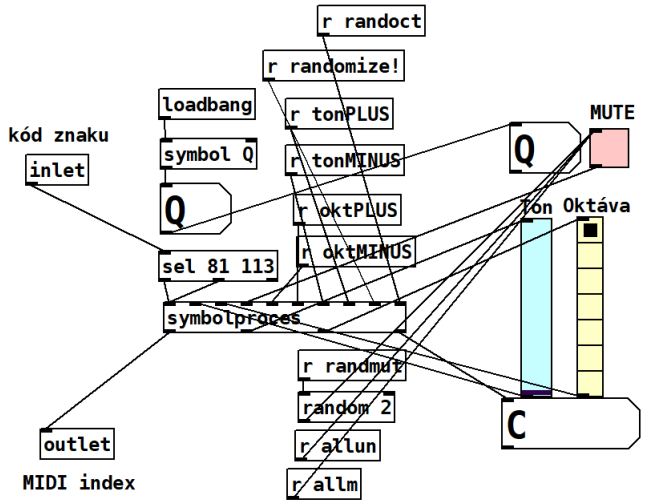
Obr. 3.12: Určení symbolu noty

`random` s argumentem 2, na jehož výstupu se po přijetí „`bang`“ zprávy objeví náhodně buď 1 nebo 0. Tak je stav toggle prvku „MUTE“ zvolen náhodně a nezávisle pro každý znak. Výjimkou je znak mezery, zakódovaný ze vstupní jednotky kódem 0. Ten je na ovládacím panelu převodníku umístěn jako první a jeho toggle prvek „MUTE“ není napojen na globální ovládací prvky utlumování.

Další objekty `recieve` přijímají zprávy z ostatních globálních ovládacích prvků. Logika jejich propojení se subpatchem `symbolproces` vyplývá z 3.3.1. Kód znaku je porovnáván s argumenty objektu `select` a v případě shody je vyslána zpráva „`bang`“ na první vstup procesoru. Ovládací prvky tónu a oktávy jsou svým výstupem napojeny na druhý a třetí vstup procesoru a svým vstupem na jeho druhý a třetí výstup. Objekty typu symbol jsou viditelné v hlavním patchi `Text_Reader`. Horní objekt ukazuje název znaku, dolní je napojen na čtvrtý výstup procesoru a ukazuje symbol zvoleného tónu. Pro přehlednost skutečného rozmístění je objekt typu symbol, indikující název znaku, přítomný dvakrát (jeden nad ovládacími prvky, druhý nad procesorem, zřetelné z obrázku 3.13).

V případě, že kód znaku neodpovídá žádnému z objektů `select`, předřazených před první vstupy subpatchů `symbolproces`, na výstup převodníku neprojde žádná zpráva. To se na výstupu projeví jako pomlka o příslušné délce kroku sekvenceru.

Na obrázku 3.16 je zpřehledněné zapojení výběru tónu pro funkci „Randomize“. Vyobrazeno je pouze řízení tónu F (ohraničeno červeně). Tato část je pro každý tón a k němu připadající ovládací prvek shodná. V případě zaškrtnutí pole toggle prvku uživatelem je na jeho výstup vyslána hodnota 1. Objekt `trigger` nejprve zaškrtne i sdružený toggle prvek (vpravo). Poté je hodnotou 1 aktivován první výstup



Obr. 3.13: Skutečné uspořádání znakových jednotek

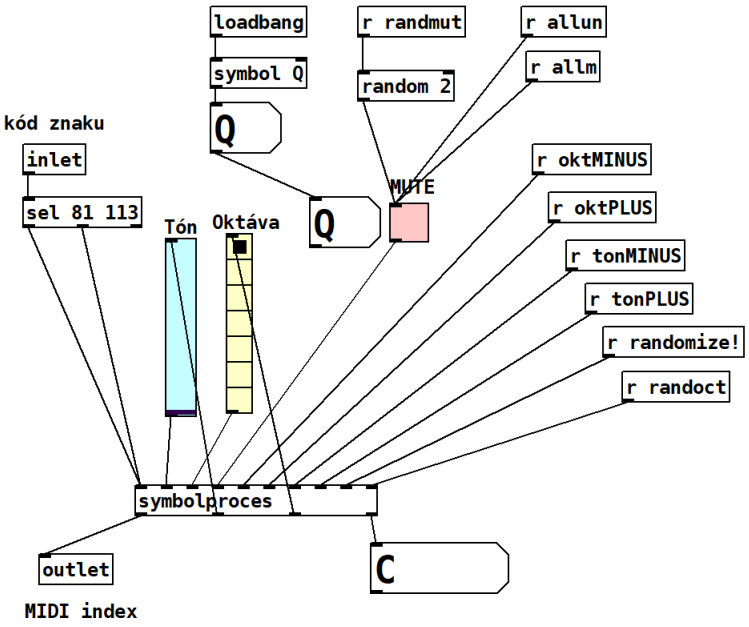
objektu `select`. Ten vyšle zprávu „append“ s argumentem odpovídajícím MIDI indexu daného tónu do objektu `list store`. Seznam v objektu je tak rozšířen o index zvoleného tónu.

V případě deaktivace ovládacího prvku výběru tónu je rovněž nejprve deaktivován sdružený toggle prvek. Následně je hodnotou 0 zvolen druhý výstup objektu `select`. Ten je napojen na `send` objekt s argumentem „`nula!`“, který v globální části aktivuje objekt `trigger`. Ten nejprve vyšle do pravého vstupu objektu `list store` zprávu „`bang`“, která vymaže jeho obsah. Pak vyšle další zprávu „`bang`“ do **všech částí náležejících jednotlivým tónům**. Ta aktivuje zprávu „`append`“ k doplnění indexu do seznamu. Nejprve však prochází skrze logickou bránu `spigot` řízenou sdruženým prvkem toggle. Do prázdného seznamu jsou tím nanovo vepsány indexy těch tónů, které jsou aktuálně vybrány.

Prvky „Vybrat: Nic“ a „Vybrat: Vše“ vysílají hodnoty 1 nebo 0 na vstupy všech toggle prvků. Tím je provedena stejná činnost, jako by uživatel aktivoval/deaktivoval všechny prvky určené k výběru tónu ručně. Seznam je vyslán do všech procesorů znaku prvkem „RANDOMIZE“.

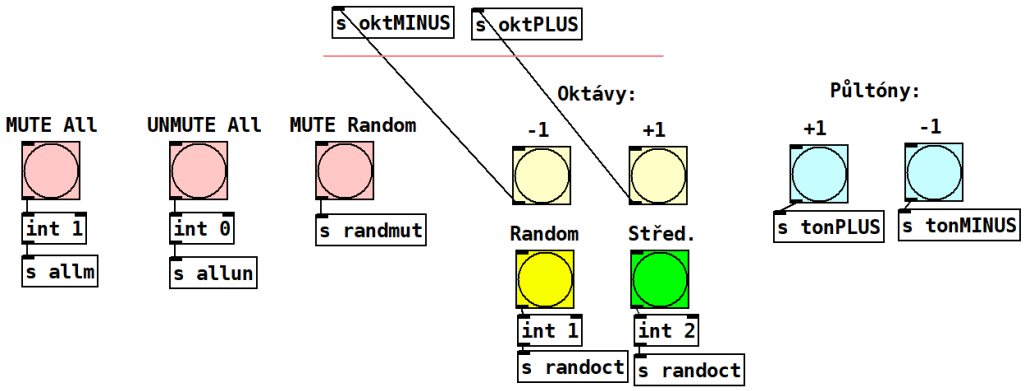
3.4 MIDI výstup

Na obrázku 3.17 je vidět výstup patche `TEXT_Reader` napravo od vstupní jednotky. Objekt `recieve` s argumentem „`conOUT`“ přijímá MIDI index na výstupu převodníku. Ten je přes logickou bránu `spigot` řízenou prvním výstupem sekvenceru přiveden na levý vstup objektu `makenote`. Prostřední vstup dále nastavuje sílu úderu

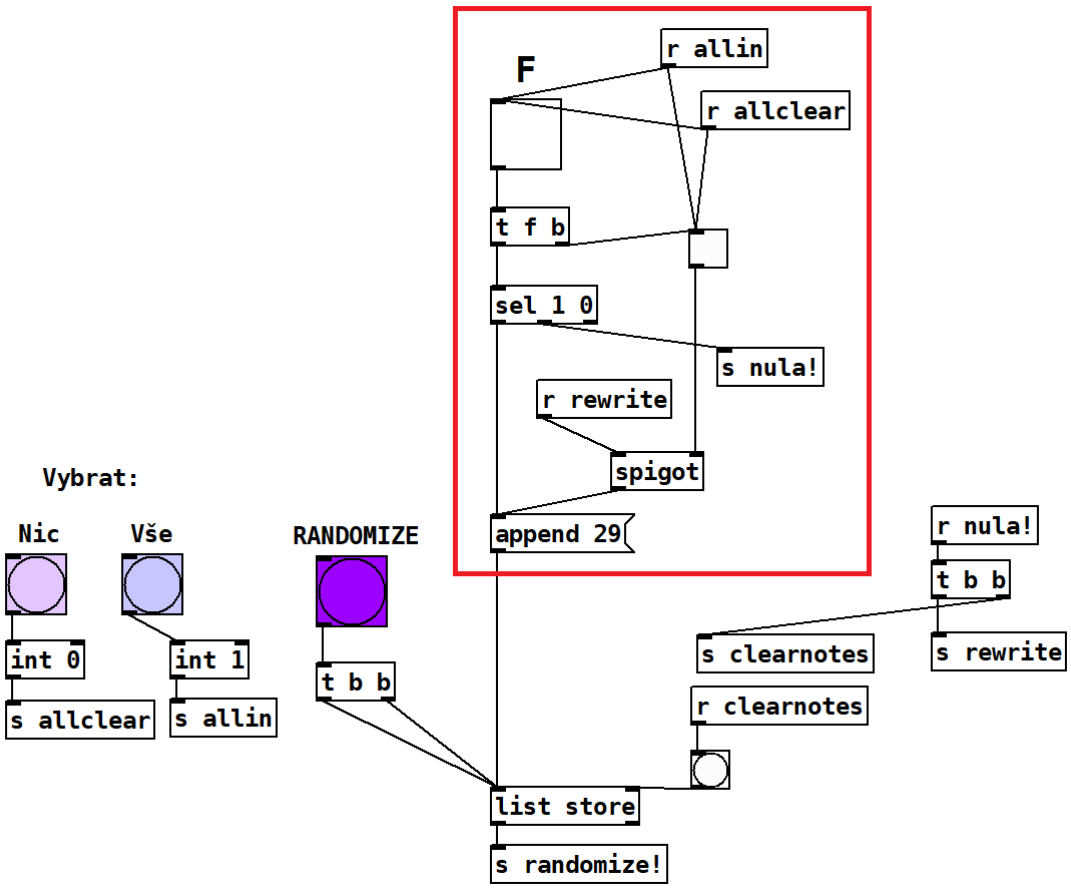


Obr. 3.14: Přehlednější uspořádání znakových jednotek

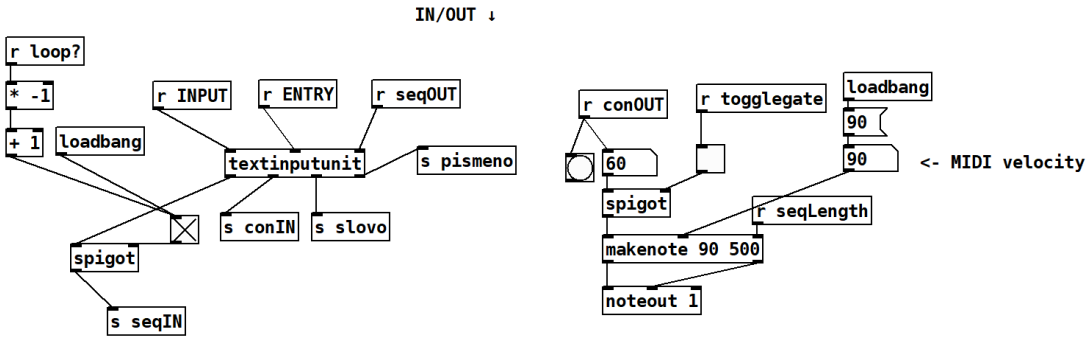
(*velocity*) v rozsahu 0–127. Pravý vstup určuje délku tónu v ms a je na něj přiveden třetí výstup sekvenceru. Objekt `noteout` vysílá na MIDI výstup Pd MIDI zprávy. Na jeho první vstup je přiveden MIDI index tónu, na jeho druhý síla úderu, kterou v reálném čase předává na výstup. Jí odpovídající číslo objekt `makenote` vysílá tak dlouho, jak je nastaveno jeho prostředním vstupem. Třetí vstup objektu `noteout` nastavuje index MIDI kanálu. Tento vstup není zapojen, místo toho je argumentem objektu při inicializaci nastaven první kanál.



Obr. 3.15: Připojení ovládacích prvků převodníku



Obr. 3.16: Znázornění funkce výběru náhodných tónů



Obr. 3.17: Vstupní jednotka a výstup patche

4 Omezení patche a jeho případné rozšíření

4.1 Podporované znaky

Patch `TEXT_Reader` dokáže přečíst 41 znaků české abecedy bez rozlišení velkých a malých písmen, znak mezery a interpunkční znaky tečka, vykřičník a otazník. Nepodporované znaky interpretuje jako pomlku o délce příslušného kroku. Přirozeně se nabízí otázka, proč jsou programem podporovány právě tyto znaky, a proč jiné znaky podporovány nejsou.

Z programátorského hlediska není kopírování jednotky zpracovávající individuální znak náročným úkonem, rozlišení velkých a malých písmen tak mohlo být implementováno snadno. Separátním zpracováním velkých a malých písmen by program rázem nabyl dalších jednačtyřiceti možných vstupních informací. Bylo však vyhodnoceno, že tato dodatečná komplexnost by pro uživatele působila velikostí ovládacího panelu jako zastrašující a ubírala by programu na eleganci. Navíc by byla svoji uměleckou podstatou mělká.

Snadno mohly být také implementovány další znaky kódované normou ASCII, jako například „$“. Vzhledem k jejich četnosti ve většině běžných textů však jejich zahrnutí mezi podporované znaky nutně nestojí za další rozšiřování ovládacího panelu a důslednou komplikaci orientace v něm. Nepodporovanými znaky s mnohem větší četností využívání jsou však čárky a číslice.

4.1.1 Absence čárek v seznamu

Jak již bylo uvedeno v 1.1.4, čárka v Pd znamená ukončení zprávy. Zahrnutí tohoto symbolu v jakékoliv datové informaci v Pd je důsledkem toho téměř nemožné. Objekt `textfile`, který v rámci subpatche vstupní jednotky `textinputunit` načítá textový soubor, tento soubor chápe jako sekvenci zpráv oddělených čárkami. Proces extrakce celého textu do jednoho seznamu je popsán v 3.1.1. Absencí čárek ve výsledné informaci vznikají hned dvě nepříjemné „nesymtrie“. Tou první je, že nemohou být podporovány všechny interpunkční znaky, pouze některé. Z tohoto důvodu bylo zvažováno jejich úplné vynechání z podporovaných znaků. Ani tím by však nezanikla nesymetrie druhá: zatímco všechny ostatní nepodporované znaky se zobrazují na indikátorech a jsou interpretovány jako pomlka o délce aktuálního kroku sekvenceru, čárky nejsou v informaci přítomny vůbec. Nejsou tedy pomlkami a nezobrazují se na indikátorech.

4.1.2 Číslice jako položky seznamu

V cestě k implementaci čtení číslic stojí podstatná překážka: Ačkoliv číslice obsažená v textovém řetězci je Pd chápána jako symbol (např. v řetězci „`m3lanch0l1k`“), pokud je číslice nebo řetězec číslic samostatnou položkou seznamu, Pd je čte jako číslo. Převedení celého čísla na řetězec symbolů odpovídající jeho číslicím je v Pd značně náročným úkonem. Je potřeba provést postupnou iteraci tohoto čísla po mocninách deseti. I tomu však stojí v cestě jistá překážka. V případě patche, který slouží ke čtení textu, si lze snadno představit situaci, kdy se uživatel rozhodne na vstup přivést textový soubor obsahující pouze jeden dlouhý řetězec číslic a pracovat tak „digitálně“ s deseti možnými tóny. Textový soubor může být téměř neomezeně dlouhý, přivedením příliš velkého čísla by v tom případě iterace nemusela být výpočetně proveditelná. Nemluvě o tom, že zadané číslo je v Pd vždy zpracováváno jako číslo s plovoucí řádovou čárkou a nemusí tak náležet celým číslům. Čísla v Pd lze navíc zadat i ve formátu s mocninami deseti, např. „`1.2e+11`“ jako reprezentace čísla $1,2 \cdot 10^{11}$. V takových případech iterační metodu provést nelze.

Přestože čtení číslic uvnitř textových řetězců je teoreticky přípustné (jako v příkladě „`m3lanch0l1k`“), procesory a ovládací prvky číslic nebyly obsaženy, aby uživatel nebyl nabádán k zadávání znaků, které jsou podporovány jen částečně. V případě výskytu čísla v čtené informaci indikátor slova zobrazí textový řetězec „`float`“. Podobně jako u nepodporovaných znaků nastane pomlka o délce aktuálního kroku sekvenceru.

4.2 Dynamika a kanály

Patch vysílá MIDI zprávy v prvním MIDI kanálu. Číslo kanálu lze pomocí *edit* módu změnit v argumentu objektu `noteout`. Zůstává však statické a stejné pro všechny tóny. Síla úderu je určena na druhém vstupu objektu `makenote`. Rovněž ji jde změnit, ale zůstává na této nastavené hodnotě. Jedním z možných rozšíření funkcí programu by mohly být možnosti přiřazení různých znaků do různých MIDI kanálů a nastavení různé síly úderu. K tomu by bylo zapotřebí rozšířit subpatch `symbolproces`, přidat převodníku `asciicon` více výstupů a umístit ke všem znakům více ovládacích prvků..

4.3 Real-time výstup

Výstup MIDI z patche `TEXT_Reader` probíhá v reálném čase za chodu sekvenceru. To na jednu stranu umožňuje provádět změny ovládacích prvků, nebo dokonce přepínat mezi různými čtenými soubory, v průběhu interpretace. Na druhou stranu pro uložení MIDI zpráv z patche je třeba jeho výstup v reálném čase zaznamenat v nahrávacím

programu. Uživatel tak potřebuje věnovat svůj čas nahrávání, pokud chce MIDI zprávy uložit. Výstup je deterministický, se stejným nastavením ze stejného čteného textu vždy generuje tutéž sekvenci.

Závěr

V rámci studentské práce byl navržen a sestaven softwarový nástroj určený ke generování hudby z textových řetězců. V textu práce byly představeny nutné teoretické poznatky. Dále byl odůvodněn návrh nástroje, zejména výběr platformy, způsobů ovládání a výstupu. Následně byl popsán způsob obsluhy nástroje a jeho architektura. Omezení nástroje popsané v poslední kapitole mohou být podnětem k navazující práci.

Nástroj vychází ze zpracování nehudební informace v podobě psaného textu, kterou konvertuje na hudební informaci v podobě sekvence MIDI zpráv. Hodnota nástroje jako prostředku k tvorbě experimentální hudby je ilustrována přiloženou skladbou. Naprogramované patche jsou současně příspěvkem do širší sféry Pure Data projektů. Části nástroje mohou být převzaty a využity v budoucích programech, patche i vzorové textové soubory jsou zpřístupněny na platformě GitHub.

Literatura

[1] KREIDLER, Johannes. *Programming Electronic Music in Pd.* [online], 2009, ISBN-13 978-3936000573. Dostupné z URL: <http://www.pd-tutorial.com/english/index.html>.

[2] YOUNG, Rob. *The MIDI files.* 2nd ed. London: Prentice Hall, 2001, xxi, 7. s. ISBN 0-130-60863-7

[3] GEIST, Bohumil. *Akustika – jevy a souvislosti v hudební teorii a praxi* Praha: nakladatelství Muzikus, 2005, 132 s. ISBN 80-86253-31-7

[4] SCHIMMEL, J. *Studiová a hudební elektronika.* Brno: Vysoké učení technické v Brně, 2012. ISBN: 978-80-214-4452- 2

[5] BROOKSHEAR, J. Glenn. *Informatika.* dotisk 1. vyd. Brno: nakladatelství Computer Press, 2013. Autorizovaný překlad z originálního vydání *COMPUTER SCIENCE: AN OVERVIEW, 11th ed.* Překlad: GONER, Jakub. 47-49 s ISBN 0132569035

[6] HEROUT, Pavel. *JAVA, grafické uživatelské prostředí a čeština.* České Budějovice: nakladatelství Kopp, 2004, 249 s. ISBN 80-7232-237-0

[7] ERICHSEN, Tobias. LoopMIDI. [online], 2022, Dostupné z URL: <https://www.tobias-erichsen.de/software/loopmidi.html>.

[8] BALWYN a INGOX. Textentry – a single line text entry box. [online], In: PURE DATA forum~, 2019, Dostupné z URL: https://forum.pdpatchrepo.info/topic/12401/textentry-a-single-line-text-entry-box.

A Příloha

Tab. A.1: Názvy tónů dle rozdílných norem

středoevropská	anglická
C	C
Cis/Des	C#/Db
D	D
Dis/Es	D#/Eb
E	E
F	F
Fis/Ges	F#/Gb
G	G
Gis/As	G#/Ab
A	A
B	**A#/Bb**
H	**B**

Tab. A.2: Délky not nabízené sekvencerem

Nota	Délka vůči čtvrťové
půlová	$\cdot\ 2$
čtvrťová s tečkou	$\cdot\ 1,5$
čtvrťová	$\cdot\ 1$
osminová s tečkou	$\cdot\ 0,75$
osminová	$\cdot\ 0,5$
šestnáctinová	$\cdot\ 0.25$
triola	$/\ 3$
kvintola	$/\ 5$
septola	$/\ 7$

B Obsah elektronické přílohy

Obsahem elektronické přílohy jsou soubory patchů `.pd`, textové soubory a zvukový soubor ukázkové kompozice. Ke spuštění nástroje je nutné mít na zařízení nainstalovaný program Pure Data (`https://puredata.info/downloads`). Soubory ve složce `Pd` tvoří funkční část nástroje. Ke spuštění slouží hlavní patch `TEXT_Reader`.

K propojení s externím hradwarovým nástrojem, zvukovou kartou, virtuálním nástrojem, nebo DAW programem je třeba správně nastavit MIDI výstup Pure Data. Pro propojení s DAW může být velmi nápomocná aplikace LoopMIDI.

- `/` ..... kořenový adresář přiloženého archivu
  - `Pd` ..... Patche a textové soubory
    - `asciicon.pd`
    - `input.txt`
    - `poet.txt`
    - `symbolproces.pd`
    - `TEXT_Reader.pd`
    - `textentry.pd`
    - `textinputunit.pd`
    - `textseq.pd`
  - `TEXT_ukazka.mp3` ..... Demonstrativní skladba