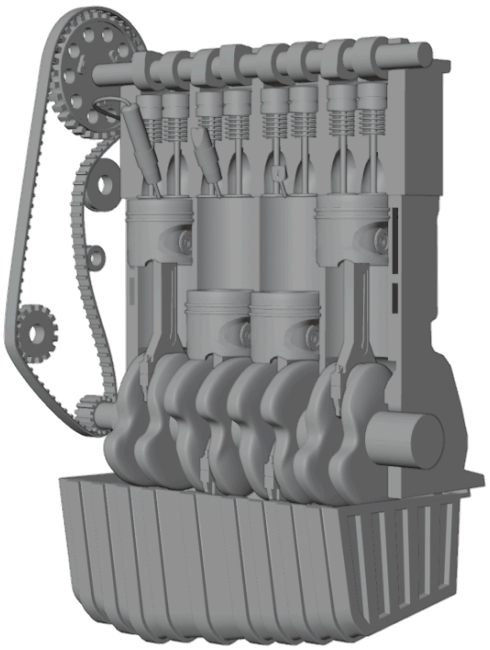
GYMNÁZIUM, PRAHA 6, ARABSKÁ 14



Adess – umělecky dirigovaný syntetizér zvuku motorů

Počet slov: 1082, normostran: 1082

Gymnázium, Praha 6, Arabská 14
tel.: 235 351 708
fax.: 222 262 066
e-mail: ga@gyarab.cz
www.gyarab.cz

Zadání ročníkového projektu

Název: **Adess – umělecky dirigovaný syntetizér zvuků motorů**

Řešitel: **Kubota Leon, 4.E**

Vedoucí práce: **Ing. Daniel Kahoun**

Datum odevzdání: **28. 2. 2026**

Způsob zpracování a kritéria hodnocení:

Zpracování v požadovaném rozsahu se řídí obecně závaznými pokyny zpracování ročníkových projektů. Řešitel elektronicky odevzdáve stanoveném termínu dokumentaci, prezentaci, poster a další vyžádané přílohy (např. zdrojové kódy, ukázková data). Před obhajobou řešitel odevzdá jeden výtisk stejné dokumentace s podepsaným prohlášením o autorství a jeden poster, neurčí-li vedoucí jinak. Hodnotí se odborné zpracování úlohy, použití návrhových vzorů, prezentace při obhajobě a funkcionalita produktu.

Popis (povinná část):

Vytvořte konzolovou aplikaci pro procedurální syntézu zvuků motorů určenou pro film a animaci. Program generuje zvukové vzorky na základě uživatelské konfigurace a klíčových snímků definovaných ve vlastním datovém formátu *DST*.

Upřesnění zadání:

- Implementace parseru konfiguračních souborů *adess*
- Procedurální generování pole vzorků zvuku motoru
- Práce s klíčovými snímky pro změnu zvuku v čase
- Export *audia* do standardního formátu *WAV*
- Ovládání aplikace pomocí příkazové řádky (*CLI*)

Bonus (nepovinná část):

- Podpora pro různé typy motorů (vrtulové, spalovací)
- Přidání zvukových efektů (např. ozvěna, zkreslení)
- Grafické znázornění generované zvukové vlny v *CLI*
- Možnost reálného náhledu (přehrání) před uložením

Platforma:

- C

datum podpisu	podpis řešitele

Anotace

Adess je aplikace přístupná v příkazové řádce, která procedurálně generuje zvuky spalovacích motorů pro využití v animaci a filmu. Generace je plně ovladatelná uživatelem prostřednictvím nastavovacích souborů *adess* ve vlastním „jazyce“ *DST* (datová ukládací věc). V těchto souborech jsou uloženy důležité hodnoty o formátu a charakteristice požadovaného zvuku a klíčové snímky, které určují jeho přeměnu v čase. Hodnoty z těchto souborů jsou načteny do paměti a následovně využity pro procedurální paralelní syntézu zvuků motorů. Výstupem této generace je pole vzorků, které se uloží do souboru *WAV*.

Abstract

Adess is an application that runs in the terminal, it procedurally generates the sound of combustion engines for use in animation and film production. The generation is fully customizable by the user through „adess“ configuration files in a proprietary DST „language“ (data storage thing). These files contain important values about the format and characteristics of the desired sound and keyframes, that determine the sounds change throughout time. The values from these files are loaded into memory and subsequently used for procedural parallel synthetization of engine sounds. The output of this generation is a buffer of samples, which are exported into a *WAV* file.

Zusammenfassung

Čestné prohlášení

Prohlašuji, že jsem jediným autorem tohoto projektu, všechny citace jsou řádně označené a všechna použitá literatura a další zdroje jsou v práci uvedené. Tímto dle zákona 121/2000 Sb. (tzv. Autorský zákon) ve znění pozdějších předpisů uděluji bezúplatně škole Gymnázium, Praha 6, Arabská 14 oprávnění k výkonu práva na rozmnožování díla (§ 13) a práva na sdělování díla veřejnosti (§ 18) na dobu časově neomezenou a bez omezení územního rozsahu.

datum podpisu	podpis řešitele

Obsah

1 Úvod

Tvorba zvuku motorů pro využití ve animaci je velice složitá, konvenčním způsobem je nahrát zvukové stopy skutečného motoru a pomocí komplexních digitálních manipulací získat finálnú zvuk. Takováto tvorba zvuku je velice časově náročná, umožňuje však precizní úpravu zvuku pro tvorbu působivých výsledků. Tento způsob je však zcela nevyhovující menším studiům či jednotlivcům, kteří nedisponují stovky hodin a neovládají tuto tvůrčí disciplínu. [1]

Druhým postupem je zvukovou stopu generovat, toho lze docílit simulací tlakových vln v motoru nebo syntézou, tedy tvorbou zvuku pomocí matematických algoritmů. V této práci využívám pro generaci zvuku spalovacích motorů syntetický přístup.

2 Použité technologie

Celý projekt je napsán v programovacím jazyce *C* (konkrétně *C99*) pro jeho výpočetní účinnost a možnost bližšího kontaktu s *hardwarem*. Tento jazyk jsem zvolil také kvůli svému zájmu o studium letectví a kosmonautiky, kde se hojně využívá pro programování *embedded* systémů.

Pro zkompilování jsem využil *CMake*, který podporuje mnoho *compilerů*. Osobně jsem zvolil běžně používaný *compiler GCC* (*GNU Compiler Collection*). [2, 3]

Již zmíněné technologie jsou vše, co je nutné pro zkompilování a spuštění projektu (uvažuji, že standardní knihovny uživatel má). Při psaní zdrojového kódu jsem však využil několika dalších nástrojů:

- *Sonic Visualiser* – *open-source* nástroj pro analýzu zvuku, umožňuje spektrální a vlnovou analýzu. [4]
- *Spek* – jednodušší, zato rychlejší a uživatelsky přívětivější nástroj pro analýzu zvuku. [5]
- *Valgrind* – nástroj běžící v příkazové řádce pro analýzu paměti programu za běhu, je velice užitečný pro zamezení únikům paměti

3 Zvuk

Abychom dokázaji zvuk tvořit, je potřeba mu alespoň povrchově porozumět. Jako zvuk označujeme vlnění částic vzduchu. Čím rychleji tyto částice kmitají (jejich frekvence), tím vyšší tón slyšíme. Čím větší je amplituda vlnění, tím hlasitější zvuk slyšíme. Jako lidé slyšíme zvuky o frekvencích od 20 až 20000 Hz a hlasitosti alespoň 0 dB. [6]

3.1 Digitální zvuk

Zvuk je ukládán jako pole vzorků, které určují, v jaké poloze se má nacházet oscilátor v reproduktoru. V tomto projektu je využit formát *WAV*.

V jednotlivých vzorcích je tedy uložena amplituda v jednotlivém čase. Datový typ vzorku se liší podle rozlišení vzorků (anglicky *bit depth*), běžně nabývá hodnot 8, 16, 24 a 32. Má aplikace dokáže exportovat v těchto *bit ratech* s využitím datových typů `uint8_t` pro *8-bit* audio, `int16_t` pro *16-bit* audio a `float` pro *32-bit* audio. Pro *24-bit* audio jsem zvolil zabalení do tří `uint8_t` místo jednoho `int32_t` s nulovým vycpáváním.

Sample rate označuje počet vzorků zaznamenaných na každou sekundu. Obvykle se používá hodnota 44100, jelikož s ní dokážeme zaznamenat celé spektrum lidsky slyšitelného zvuku. Při příliš nízkých *sample ratech* dochází k *aliasování*, což je nežádoucí artefakt plynoucí z příliš vysoké frekvence. Tato konkrétní frekvence, při níž dochází k *aliasování*, se nazývá *Nyquistova*. Je rovna polovině *sample ratu*. V mé aplikaci si uživatel může zvolit libovolný *sample rate*.

4 Příkazy

5 Ukládání dat

Data se v *Adess* ukládají do souborů

6 Postup *renderování* zvuku

Uživatel pomocí příkazu `adess render` se jménem scény jako argument spustí několikafázový proces *renderování*. Ten je podrobně popsán v následujících podkapitolách.

6.1 Čtení vstupních dat

Nejprve se otevře soubor projektu a jeho data se po kontrole syntaxe parsují do struktury `struct Project`. Poté se toto opakuje se scénou a nakonec se souborem motoru.

6.2 Předvýpočetní fáze

V této fázi se předvypočítají důležitá pole pro následující fáze. Tyto výpočty probíhají paralelně.

6.2.1 Interpolace klíčových snímků a z nich plynoucích hodnot

První vlákno lineárně interpoluje klíčové snímky pomocí rovnice (1), výstupem je pole frekvencí, pole otáček, pole fází, pole zátěže a pole násobitelů nízkofrekvenčního šumu.

$$h = h_0 + (t - t_0) \cdot \frac{h_1 - h_0}{t_1 - t_0} \tag{1}$$

h je okamžitá hodnota, h_0 a h_1 jsou hodnoty předchozího a následujícího klíčového snímku
t je čas, t_0 a t_1 jsou časy předchozího a následujícího klíčového snímku

Otáčky jsou vypočítány pomocí otáčkové rovnice (2).

$$ot = \frac{f \cdot 60}{n} / v \tag{2}$$

ot jsou otáčky [ot · s^{-1}]
f je frekvence [Hz]
n je inverzní počet pracovních dob za sekundu (2 pro čtyřdobé motory a 1 pro dvoudobé)
v je počet válců

Fázi vypočítáme pomocí rovnice (3).

$$\varphi_n = \sum_{i=0}^{n-1} \tau \cdot f[i] \cdot \Delta t \tag{3}$$

V programu vypočítáme rovnici (3) takto:

```
double phase = 0; // Musí být double, jelikož float neposkytuje dostatečnou přesnost

while (i < scene->sampleCount) {
  phase += TAU * frequencyBuffer[i] * timeStep;
  phaseBuffer[i] = phase;
  i++;
}
```

Násobitel nízkofrekvenčního šumu vypočítáme pomocí rovnice (4).

$$n = s \cdot \left(\frac{-ot^2 - 2 \cdot ot \cdot ot_v}{p} - \frac{ot_v}{p} + 1 \right) \tag{4}$$

n je násobitel nízkofrekvenčního šumu
s je síla šumu
ot a ot_v jsou okamžité otáčky a otáčky ve volnoběhu
p je pokles (vzdálenost od volnoběhu ve které je šum nulový, v otáčkách)

6.2.2 Generace stabilního hnědého šumu

6.2.3 Generace nízkofrekvenčního šumu

6.3 Výpočetní fáze

6.4 Kombinační fáze

6.5 *Post-processing* fáze

7 Závěr

8 Seznam použitých zdrojů

[1] JACKSON, Blair. Engine FX With Personality in Pixar's Cars. *Mix* [online]. 2006 [vid. 2026-01-14]. Dostupné z: https://www.mixonline.com/sfp/engine-fx-personality-pixars-cars-369139

[2] KITWARE, INC. *CMake* [online]. 2026 [vid. 2026-01-18]. Dostupné z: https://cmake.org/

[3] FREE SOFTWARE FUNDATION. *GNU Compiler Collection (GCC)* [online]. 2026 [vid. 2026-01-18]. Dostupné z: https://gcc.gnu.org/

[4] CANNAM, C., C. LANDONE a M. SANDLER. Sonic Visualiser: An Open Source Application for Viewing, Analysing, and Annotating Music Audio Files. In: *Proceedings of the ACM Multimedia 2010 International Conference*. 2010, s. 1467–1468.

[5] KOJEVNIKOV, Alexander. *Spek – Acoustic spectrum analyzer* [online]. 2025 [vid. 2026-01-18]. Dostupné z: https://www.spek.cc/

[6] DOSITS. [online]. 2025 [vid. 2026-01-18]. Dostupné z: https://dosits.org/science/measurement/what-sounds-can-we-hear/