Fakulta elektrotechnická ČVUT v Praze
Obor Softwarové inženýrství a technologie

Semestrální práce

Dokumentace

Filip Růžička
ruzicfi3

Leden, 2025

Obsah

1. Úvod

Tato semestrální práce dokumentuje vývoj klasické arkádové hry Pong v jazyce C++. Cílem projektu bylo nejen naučit se v tomto programovacím jazyce, ale také vytvořit plně funkční hru, která slouží jako demonstrace získaných vědomostí.

V rámci práce jsem se zaměřoval na správné použití vláken, získávání vstupu od uživatele a řešení jednoduchých algoritmů, jako třeba pohyb počítačem řízeného protihráče.

2. Implementace

Práce se drží principů objektově orientovaného programování (OOP). Program je navržen tak, aby byl modulární, což usnadňuje ladění i případné budoucí rozšiřování (např. o různé úrovně obtížnosti). Pro vývoj byl zvolen standard C++20, který nabízí moderní nástroje pro práci s vlákny a atomickými operacemi.

Jelikož hra běží v terminálu bez externích grafických knihoven, vykreslování probíhá pomocí ANSI únikových sekvencí (ANSI escape codes), které umožňují mazat obrazovku a přesouvat kurzor na konkrétní souřadnice $[x, y]$. Základem celého programu je herní smyčka (game loop), která běží s pevnou frekvencí (tick rate).
Ta v každém kroku provádí následující operace:

1. Zpracování uživatelského vstupu z fronty.
2. Aktualizace stavu herních objektů (pohyb míčku, pohyb pálky počítače).
3. Detekce a řešení kolizí.
4. Překreslení scény do terminálu.

2.1. Struktura tříd

Program je strukturován do několika tříd:

- `paddle` – reprezentuje pálku, obsahuje metody pro pohyb nahoru a dolů (`moveUp()`, `moveDown()`), vykreslení (`draw()`) a získání pozice (`getRect()`).
- `ball` – reprezentuje míček, obsahuje například metody pro pohyb (`move()`), vykreslování (`draw()`), detekci kolizí s pálkami a stěnami (`checkCollision()`, `checkGoalCollision()`), a tak dále.
- `game` – řídí herní logiku, zpracovává vstupy od hráče, aktualizuje pozice pálky a míčku, vyhodnocuje skóre a vykresluje stav hry v terminálu.

2.2. Vlákna

Pro zajištění plynulosti hry využívá aplikace dvě paralelní vlákna. Toto rozdělení řeší hlavní limitaci terminálových aplikací – blokování programu při čekání na uživatelský vstup.

Vlákno pro vstup (Input Thread): Běží v nekonečné smyčce a asynchronně zachytává stisky kláves. Tyto stisky přidává do fronty, ze které pak čte druhé vlákno. Díky tomu není hlavní herní logika blokována čekáním na uživatelský vstup.

Vlákno pro logiku a vykreslování (Compute Thread): V pravidelných intervalech (tick rate) přepočítává pozice objektů, detekuje kolize a následně překresluje scénu do terminálu.

Zvažoval jsem separaci výpočtů a vykreslování do dvou samostatných vláken (tedy celkem 3 vlákna). Od tohoto řešení jsem však upustil, protože v rámci terminálové hry jsou tyto operace lineárně závislé – tyto dvě vlákna by se pouze střídala.

Synchronizace vstupní fronty je řešena pomocí mutexu a booleanová hodnota `stop` je atomická, aby nedošlo k race condition.

3. Kompilace a spuštění

Projekt využívá sestavovací systém CMake, který automatizuje generování Makefile a usnadňuje správu závislostí. K úspěšné kompilaci je vyžadován kompilátor podporující standard C++20 a nainstalovaný nástroj CMake (verze 4.0 nebo vyšší).

Pak jde program sestavit a spustit pomocí těchto příkazů:

```
$ cmake -S . -B build     # vytvoření build složky
$ cmake --build ./build   # samotné sestavení
$ cd build && ./pong      # spuštění
```

3.1. Popis přepínačů (argumentů)

Program podporuje následující přepínače:

`--help` – zobrazí nápovědu s popisem ovládání a dostupných přepínačů.

`--width <val>` – nastaví šířku hracího pole na hodnotu `<val>`.

`--height <val>` – nastaví výšku hracího pole na hodnotu `<val>`.

Příklad použití:

```
$ ./pong --help

$ ./pong --width 100 --height 30
```

3.2. Ovládání aplikace

Hráč ovládá svou pálku pomocí šipek ↑ a ↓, nebo alternativně pomocí kláves W a S. Cílem je odrazit míček tak, aby soupeř jej nezachytil a získat tak bod.

Druhý hráč je počítač, který automaticky reaguje na pozici míčku. Není nijak zvlášť chytrý, protože jen hloupě následuje vertikální pozici míčku. Má však omezenou rychlost a tak je možné ho porazit.

4. Testování programu

Bylo provedeno testování na více scénářích (nastavení hry bylo defaultní):

Scénář	Vstup	Očekávaný výstup	Výsledek
Pozice $[y]$ pálky: 5	W	Pozice $[y]$ pálky: 4	OK
Pozice $[y]$ pálky: 5	S	Pozice $[y]$ pálky: 6	OK
Pozice $[y]$ pálky: 0	W	Pozice $[y]$ pálky: 0	OK
Pozice $[y]$ pálky: max	S	Pozice $[y]$ pálky: max	OK
Pozice $[x, y]$ míčku: $[0, 4]$		Gól	OK
Pozice $[x, y]$ míčku: $[\text{max}, 8]$		Gól	OK
Pozice $[x, y]$ míčku: $[20, 10]$		Nic	OK

4.1. Herní strategie

Odpálit míček tak, aby ho oponent (počítač) nezvládl chytit. Míček se postupem času zrychluje, a tak dříve nebo později jednomu z hráčů proletí.

5. Závěr

Přestože byla výsledná aplikace úspěšně implementována a splňuje všechna technická kritéria, proces vývoje výrazně komplikovala nízká srozumitelnost a přehlednost jednotlivých částí zadání. Zkušenost s jazykem C++ v rámci tohoto projektu pak potvrdila mou preferenci pro jiné programovací nástroje, jelikož vzhledem k jeho nepřehledné a nejasně definované syntaxi a nadměrné množství vzájemně se překrývajících programových konstruktů pro implementaci stejné logiky, jej do budoucna nepovažuji za jazyk, ve kterém bych se chtěl dále realizovat.

Odkazy

1 ČVUT logo. https://cs.wikipedia.org/wiki/Soubor:CVUT_znak.svg.