Zápočtový program

Modifikace Hry
Super Mario

David Beinhauer

2. ročník

Programování v C++ (NPRG041)

Letní semestr 2020/21.

Anotace

Program je realizací plošinové arkádové hry inspirované hrou Super Mario. Je napsán v jazyce C++ s využitím grafické knihovny SFML. Hra obsahuje logiku pouze pro 1 úroveň, jejíž reprezentace mapy je zapsána v přidruženém textovém dokumentu.

Podrobný popis hry

Hra je založena na pohybu hráče na mapě, jehož cílem je sesbírat co nejvíce tokenů (peněz) a v bezpečí se dopravit do cíle.

Hráč začíná na dané startovní pozici a je mu umožněn pohyb po neprůchodných předem umístěných blocích. Pohyb je ovlivněn „gravitací“, takže není například možné vyskočit do nekonečné výšky a například při pádu zrychluje (resp. při skoku postupně zpomaluje). Hráč se pohybem po mapě snaží dostat do cílové pozice. Během cesty sbírá peníze (body) a přitom se vyhýbá nástrahám mapy.

Mezi nástrahy mapy patří „díry“ v mapě (místa, kde již pod hráčem není žádný blok), pokud zde hráč spadne, pak ztrácí život a vrací se na začátek. Další překážkou v cestě jsou protivníci, kteří mohou „zabít“ hráče společnou kolizí nebo střelou, která následně hráče zasáhne. Pokud ale hráč na nepřítele skočí, je nepřítel zničen a hráči zůstávají všechny životy.

Je zde také možnost vzájemného zničení nepřátel. Pokud nepřítelova střela zasáhne nepřítele (teoreticky i sám sebe), pak zasažený vystřelí odvetnou silnější střelu opačným směrem. V případě, že silnější střela zasáhne nějakého z nepřátel, je zasažený zničen.

Hra končí pokud hráč ztratí veškeré životy, nebo pokud dorazí do cílové pozice.

Popis algoritmů

Zobrazení okna

Pro zobrazení správného okna je použit algoritmus založený na detekci stavů hry *GameState* a volání vhodných funkcí pro daný stav (především pro zobrazení správných objektů, detekci vhodných akcí, ovládání logiky hry a podobně).

Stavy reprezentují následující události. Pokud je program ve stavu *STATE_START,* je zobrazeno počáteční menu s názvem hry a volbami pro začátek hry a konec programu. Dále stav *STATE_END_MENU* se téměř neliší od *STATE_START,* pouze místo názvu vypíše výsledek poslední odehrané hry. Stav *STATE_GAME* slouží pro veškerou logiku hry. Nakonec *STATE_END* je ukončovací stav, sloužící pouze jako indikace pro následné ukončení celého programu.

Hra

Samotný hrací algoritmus je založen především na kontrole překrývání vhodných objektů na hrací mapě a následná aplikace vhodných funkcí na základě druhu kolize. Významnou částí algoritmu je práce s relativními souřadnicemi sloužící k zefektivnění algoritmů pro detekci kolize.

Reprezentace mapy je načtena jako 2D pole (formát načítání a reprezentace je popsán níže v programátorské dokumentaci) a slouží jako relativní soustava souřadnic. Systém je založen na předpokladu, že každá překážka má stejný rozměr, tedy mapa lze číst jako vhodný textový soubor, kde každý znak reprezentuje políčko na mapě velikosti 1 překážky. Relativní souřadnice tedy udávají v jakém násobku (výšky resp. šířky) se nachází levý horní roh překážky (v absolutních souřadnicích hracího okna).

Zmíněná relativní reprezentace umožňuje zhruba určit polohu objektu na hrací mapě (teoreticky, i pokud je objekt větší než překážka po vhodném převodu velikosti objektu do „relativní“ velikosti). Tato skutečnost se následně využívá pro odhad pozice objektu takovým způsobem, že se vezme relativní souřadnice každého rohu objektu (objekty jsou ohraničeny obdélníkem) a ze zmíněných souřadnic se již zkontrolují statické objekty (jsou pevně na stejné pozici) pro detekci kolize (objekt se pohybuje také o „mezikroky“ mimo relativní souřadnice).

Navíc u pohybu hráče je naimplementovaná jednoduchá gravitace, která pouze aktualizuje rychlost pádu hráče v daných časových intervalech.

Programátorská dokumentace

Knihovny a sestavení programu

Program využívá grafickou knihovnu SFML. A k jeho zkompilování je tedy třeba zmíněnou knihovnu stáhnout a provézt potřebné operace (více zde https://www.sfml-dev.org/tutorials/2.5/start-vc.php).

Jednodušší varianta sestavení je použití nástroje *vcpkg* (více zde https://github.com/microsoft/vcpkg), který je ale potřeba nejprve nainstalovat (ideálně s využitím nástroje *git*). Ke správnému fungování je také potřeba program Visual Studio 2019.
Pro instalaci nástroje *vcpkg* je třeba na uživateli požadovaném místě v příkazovém řádku spustit příkazy:

```
git clone https://github.com/microsoft/vcpkg
.\vcpkg\bootstrap-vcpkg.bat
```

Následně pro nainstalování požadované knihovny a integraci do prostředí Visual Studio, je třeba ve stejném adresáři spustit příkazy:

```
.\vcpkg\vcpkg install sfml
.\vcpkg\vcpkg integrate install
```

K sestavení projektu je následně jen potřeba otevřít soubor *Platformer_game.sln* v programu Visual Studio a zde sestavit projekt. Je také třeba zmínit, že zvolený program funguje zaručeně správně pouze s 32-bitovou verzí knihovny SFML, tedy při sestavování je doporučeno zvolit možnost *x86*, namísto *x64.*

V případě neúspěšného sestavení projektu užitím vcpkg, je možné projekt sestavit způsobem popsaným v dokumentaci SFML (viz. odkaz v prvním odstavci).

Načtení mapy

Pro správné načtení mapy je potřeba textový soubor správného formátu, jehož relativní cesta je zapsána jako parametr při volání konstruktoru objektu *Game*.

Formát textového souboru je následující. Na prvním řádku se nachází 2 čísla oddělené mezerou. První číslo symbolizuje počet sloupců (*columns)* mapy (relativní šířka), 2. číslo značí počet řádků (rows) mapy (relativní výška).
Zbylé řádky se skládají z reprezentace mapy herní plochy. Ta musí být složena z *rows* řádků a každý řádek se musí skládat z *columns* sloupců povolených znaků. Řádky a sloupce překračující zmíněné počty se ignorují a je možné takto okomentovat reprezentaci mapy (znázornění souřadnic apod.).

V reprezentaci mapy jsou povoleny pouze následující znaky (reprezentující herní objekty):

- # - pevná překážky
- . - volný prostor
- P - hráč

- C - mince
- E - nepřítel
- F - konečná pozice

Je doporučeno ohraničit mapu mimo spodního okraje pomocí symbolů překážky, aby nedošlo k opuštění mapy hráčem (možné chyby v programu). Chybějící pevné symboly na dně mapy symbolizují díry v mapě a slouží k logice hry.

Je důležité zmínit, že jakékoli odchylky od výše popsané reprezentace mapy mohou vézt k nesprávnému načtení mapy, případně k špatnému chování programu.

main.cpp

Kód sloužící ke spuštění programu. Vytváří se zde herní objekt (třídy *Game*, který je následně aktualizován (v daných časových intervalech)). Nastavují se cesty k souboru reprezentace mapy a k souboru s uloženým používaným fontem. Kontroluje konec programu.

SFML_includes.h

Hlavičkový soubor obsahující includes pro všechny dll soubory knihovny SFML.

Game.cpp

Obsahuje definici třídy kontrolující veškeré akce na okně programu a řídící logiku hry. Objekt třídy obsahuje veškeré herní objekty (mapa, hráč…).

Inicializuje okno programu (nastavuje hlavní menu, nebo vykresluje hru). K nakreslení objektů slouží funkce *Game::render().* V konstruktoru objektu je možné nastavit konstantní herní parametry (vlastnosti okna, rychlosti herních objektů, časové intervaly apod.).

Game::update(sf::Clock& updateClock)

Slouží k aktualizaci stavů hry a objektů okna. Kontroluje vnější vlivy (zavření okna, vypnutí programu, stisk tlačítka menu, stisk klávesy pro pohyb hráče) a provádí vhodné akce. Parametrem je odkaz objekt třídy *sf::Clock*, sloužící pro správnou aktualizaci pohybu objektů, jenž je úměrný času od poslední aktualizace (tím je zaručen jednotný chod programu bez ohledu na výkonnost CPU).

Speciálně pro vykreslení Menu programu spravuje rozmístění objektů na okně a kontroluje volby tlačítek.

Během samotné hry spravuje pohyby hráče (horizontální pohyb, gravitační působení), protivníků (pohyb, výstřely, změny směru pohybu…) a střel, řídí také veškeré kolize objektů a kontroluje konec hry.

Ze zmíněných herních jevů funkce speciálně kontroluje časové intervaly mezi danými akcemi, případně zmíněné akce provádí a nuluje časomíru. Mezi takové akce patří náhodné generování střel, náhodná změna pohybu protivníka a aktualizace gravitačního zrychlení objektu hráče.

Level.cpp

Obsahuje definici třídy obsahující reprezentaci mapy a kontrolující veškeré kolize objektů a jejich pozice na mapě. Objekt také obsahuje kontejnery, v níž jsou uloženy veškeré herní objekty (kromě objektu hráče, jenž je obsažen ve třídě *Game* podobně jako objekt *Level*.

V konstruktoru je načtena herní mapa a jsou zde nastaveny veškeré potřebné parametry jako velikosti herních objektů a reprezentace relativních souřadnic.

Funkce tvaru Level::get…

Slouží jako přístup k herním objektům pro řízení logiky hry ve třídě *Game*. Vrací vhodné kontejnery v níž jsou uloženy dané objekty (volba kontejnerů je odvozena od používání objektů ve smyslu kontroly kolizí).

Funkce pohybující objekty (tvaru Level::move…, Level::jump…, Level::fall…)

Slouží kontrolovaný posun objektů. Jsou zde kontrolovány kolize s překážkami, přesuny mezi relativními souřadnicemi apod. Funkce vrací booleovskou proměnnou symbolizující, zda byla akce úspěšná pro propagaci informace dále.

Funkce tvaru Level::check…

Kontrolují kolize objektů (především s hráčem). Pracují s relativními souřadnicemi (podrobněji v popisu algoritmu výše). Navíc se zde vyskytující funkce pro kontrolu případného přepadu protivníků přes okraj platformy (zabránění jejich pádu mimo mapu) a také kontroly opuštění mapy (především pro pády hráče a mazání střel mimo herní mapu).

AbstractObject.cpp

Je základem všech herních objektů. Reprezentace libovolného herního objektu na mapě (hráč, překážka, mince…). Obsahuje okenní objekt reprezentující objekt. Představuje uživatelsky příjemnější prostředí pro práci s okenními objekty pro námi definované herní prostředí (vykreslení, posun, nastavení pozice, zisk hranic a souřadnic objektu). Další třídy jej využívající obsahují požadovaný objekt jako součást třídy.

LivingObject.cpp

Sloužící pro třídy pohybující se po herní mapě. Obsahuje objekt *AbstractObject* a rozšiřuje rozhraní pro práci s relativními souřadnicemi (nutné někde ukládat, nejsou v reprezentaci mapy). Dále nabízí rozhraní pro pohybování objektů.

Player.cpp

Potomek třídy *LivingObject* rozšiřující třídu o specifické vlastnosti objektu hráče (především pro správu skoků a gravitace). Nabízí možnost rozšíření o další funkce specifických pro hráčský objekt.

Enemy.cpp

Potomek třídy *LivingObject* rozšiřující třídu o specifické vlastnosti objektu protivníka (pohyb, střely). Nabízí možnost rozšíření o další funkce specifických pro hráčský objekt.

Bullet.cpp

Potomek třídy *LivingObject* rozšiřující třídu o specifické vlastnosti objektu střely (rozlišení typu, rychlost). Nabízí možnost rozšíření o další funkce specifických pro hráčský objekt.

Coin.cpp, Finish.cpp, Obstacle.cpp

Mimo vlastnictví objektu třídy *AbstractObject* slouží pouze pro nastavení okenních reprezentací herních objetků po řadě mince, finální pozice a překážky. Nabízí také možnost pro rozšíření funkčnosti objektů.

Uživatelská dokumentace

Spuštění hry

Program po spuštění zobrazí okno, na němž je vyobrazeno hrací menu, kde se vyskytují 2 tlačítka, která lze aktivovat stiskem levého tlačítka myši s kurzorem umístěným na požadovaném tlačítku. Tlačítko s nápisem *Play* spustí novou hru, tlačítko s nápisem *Exit* celý program ukončí (program lze také ukončit zavřením hracího okna nebo stiskem klávesy **Esc**}. Stejné menu je zobrazeno také po konci jedné hry, je zde ale navíc vypsáno i konečné skóre a zda hráč vyhrál, čí prohrál.

Ovládání a průběh hry

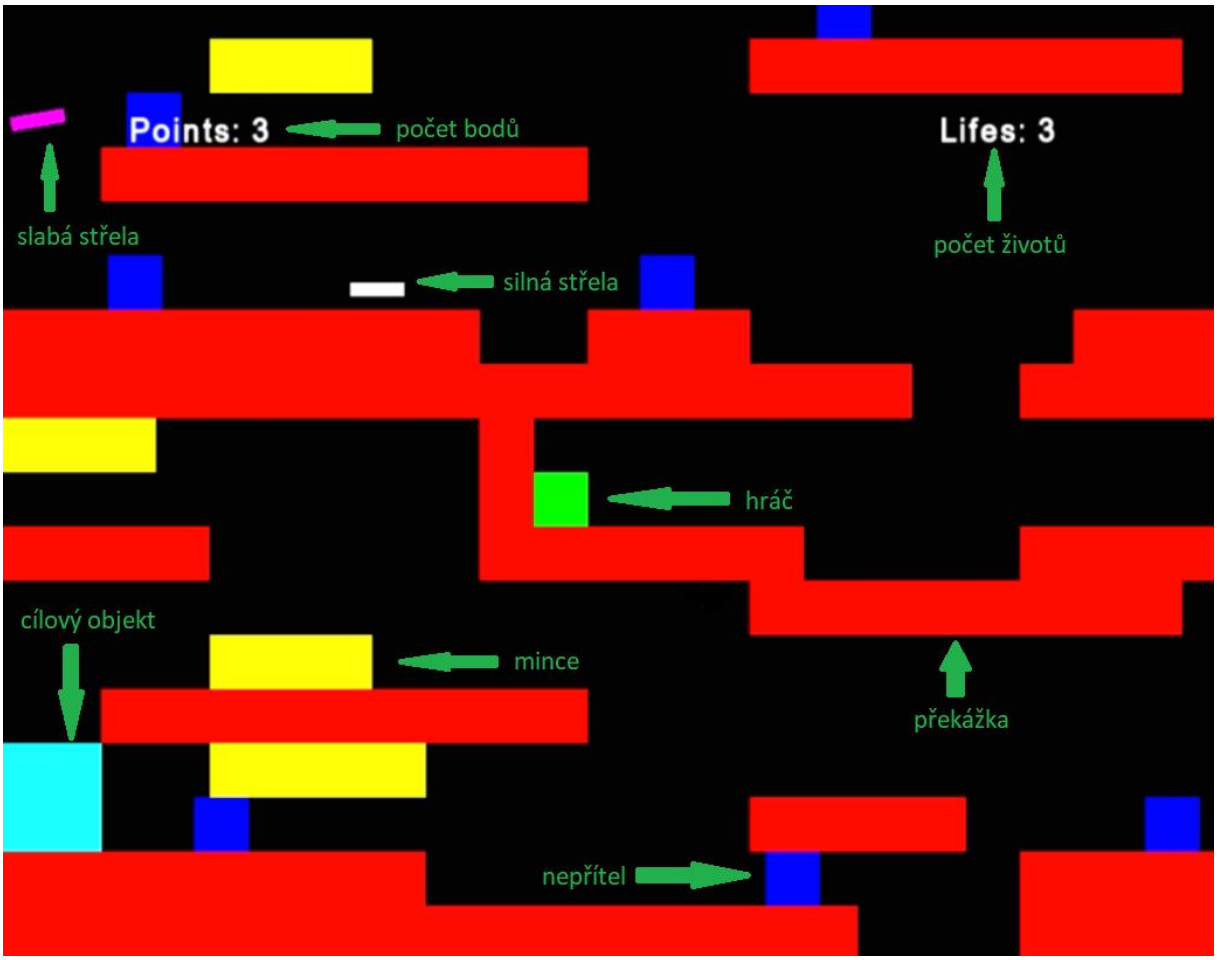
Po volbě možnosti *Play* je na hracím okně vyobrazena mapa s hracími objekty reprezentovanými obdélníky různých barev a velikostí. Reprezentace je popsána na přiloženém obrázku. Uživatel může pohybovat hráčským objektem stiskem kláves **Left**/**A** pro pohyb hráče doleva na mapě, stiskem **Right**/**D** doprava a stiskem **Up**/**W** skok hráče, který je umožněn, pouze pokud se hráčský objekt dotýká spodní částí objektu překážky. V opačném případě je ve fázi letu a vertikální pohyb je řízen plně automaticky bez možnosti vnějšího vlivu. Pohyb hráče je také omezen překážkami, pokud hráč koliduje s překážkou, není mu umožněn další pohyb kolidujícím směrem.

Hráč má také možnost sbírat mince a zvětšovat tak své bodové konto. Sběr je možný pohybem hráče tak, aby kolidoval s objektem mince. V tomto případě pak objekt mince zmizí z obrazovky a hráči je přičten odpovídající počet bodů.

Hráč je ohrožován objekty nepřátel, pokud hráč koliduje s libovolným nepřítelem jinak než pádem hráče na protivníka, bude hráči odebrán 1 život a bude navrácen do počáteční pozice. Pokud ale hráč spadne na protivníka, bude protivník zničen a zmizí z hracího pole. Nepřítel také ohrožuje hráče střelami, které v případě kolize s hráčem vedou ke stejnému scénáři se ztrátou 1 života. Pokud střela zasáhne protivníka, pak je zasaženým vystřelena odvetná silnější střela, která umožňuje zničit také nepřátelský objekt.

Hra končí v případě, že hráč dorazí do cílové pozice, nebo pokud mu nezbývají žádné další životy. Nakonec je vypsáno herní menu s výpisem výsledku a tlačítky shodnými s hlavním menu (viz. výše).

Na obrázku níže je vyobrazen obrázek s popisem herního pole.


Points: 3
počet bodů
Lifes: 3
slabá střela
počet životů
silná střela
hráč
cílový objekt
mince
překážka
nepřítel

Testovací data

Pro otestování funkčnosti programu je třeba především otestovat všechny dílčí části programu. Jejich umožňují testovací úrovně uložené v adresáři *Levels/Tests*, které mají za cíl co nejlépe připravit prostřední pro otestování specifického aspektu hry (popis testu uvnitř textové reprezentace úrovně). Také je vhodné kontrolovat správné zobrazování bodů a životů na okně a správnou funkčnost tlačítek a zobrazování hlavního menu, povaha testovaných aspektů ale umožňuje test na libovolné korektní reprezentaci úrovně.

Testovací mapy je možné spustit, pokud spustíme program s jedním argumentem, kterým je cesta k testovací reprezentaci mapy.

Příkazem ve formátu: ./PlatformerGame.exe path_to_testing_file