Semestrální práce z předmětu KIV/UUR - Jednoduchý klon hry Bomberman

Autor: Milan Vlachovký

Obsah

1. Úvod

Cílem tohoto projektu bylo vytvořit jednoduchý klon hry Bomberman s využitím webových technologií. Projekt je zaměřen na demonstraci základních principů fungovaní hry při zajištění pohodlného ovládání a zážitku pro uživatele. Projekt je koncipován jako školní semestrální práce, takže určité aspekty, které by v reálné produkci byly důležité, jsou zde zjednodušeny nebo vynechány (řádný WSGI server, ...).

2. Popis hry

Jedná se zjednodušenou verzi hry Bomberman, kde hráč ovládá postavu na obdélníkovém herním poli. Hráč má za úkol zničit všechny nepřátele a překážky na herním poli pomocí bomb, které může pokládat. Hráč má k dispozici 1 bombu, kterou může pokládat na herní pole v daný okamžik. Bomba má časovač a po uplynutí určitého času exploduje a zničí všechny entity v jejím okolí. Výbuch se ale nemůže šířit přes nezničitelné zdi v okolí. Hráč má k dispozici 0 až 3 záložních životů a pokud je zasažen nepřítelem nebo explozí, ztrácí jeden život. V této práci jsou ke hraní dostupné 2 herní módy:

- *Normal* mód
  - Hra končí, pokud hráč ztratí všechny životy nebo projde všemi levely (level může skončit, pokud jsou zničeny všechny nepřátelské entity a všechny ničitelné zdi).
- *Endless* mód
  - Hra končí, pokud hráč ztratí všechny životy. Do té doby se hráč snaží dostat do co nejvyššího levelu, nahrát co největší skóre a vše zvládnout v co nejkratším čase.

Hra je ovládána pomocí klávesnice. Hráč se pohybuje pomocí šipek a pokládá bomby pomocí mezerníku. Hra je zobrazena v okně prohlížeče a je možné ji hrát na jakémkoliv zařízení s webovým prohlížečem a klávesnicí. Hra je doplněna o zvukové efekty. Hra obsahuje žebříček nejlepších hráčů, který je ukládán na serveru a je možné se do něj zapsat po skončení hry. Výběr počtu životů se dá chápat jako obtížnost hry. Všechny použité textury a zvuky jsou z volně dostupných zdrojů a jsou použity nekomerčně.

3. Architektura systému

Projekt je rozdělen na frontendovou část, která zahrnuje klientský kód napsaný v JavaScriptu s využitím knihovny PIXI.js, a backendovou část, kterou tvoří webový server (Nginx), dokumentová databáze MongoDB a Pythonovský script využívající knihovnu Flask pro asynchronní (AJAX) zpracovávání POST a GET požadavků pro komunikaci s DB. Při připojení k serveru se na klient pošlou statické soubory (HTML, CSS, JS) a celá hra se odehrává na straně klienta. Backend slouží pouze pro ukládání skóre a získávání žebříčku nejlepších hráčů. Statické soubory posílané na klienta jsou dopředu zpracovány balíčkovacím nástrojem Parcel. **Pro jednoduchost testování a spuštění projektu byl zřízen jednoduchý server na adrese: http://34.75.224.117 kde je možné hru vyzkoušet bez nutnosti lokální instalace**.

Požadavky

- Node.js/npm
- Python 3.10 a vyšší
- MongoDB
- Nginx

4. Návod na zprovoznění

Pro sestavení statických souborů stačí spustit:

```
npm run build
```

Ten za použití Parcelu vytvoří složku *dist/* s výslednými soubory (případně *npm run build-dev*, který vytvoří soubory s mapováním pro ladění). Tyto soubory je potřeba servírovat pomocí webového serveru (např. zmiňovaný Nginx). Pro získávání a ukládání informací do DB přes AJAX volání je potřeba spustit Flask server (primitivní, nativní přes Python; nevhodný pro reálnou produkci, protože se nejedná o plnohodnotný WSGI server, ale pro účely této práce postačuje). Pro správnou funkčnost je potřeba mít spuštěný MongoDB server s defaultním nastavením pro porty. Pro spuštění Flask serveru je potřeba mít nainstalované závislosti z `requirements.txt` a spustit `db_init.py` pro inicializaci DB a následně `ajax_handler.py` pro spuštění Flask serveru. Tento script bude naslouchat na portu 5000 a bude zpracovávat požadavky z frontendu.

Instalace krok za krokem

Pro instalaci a nastavení projektu proveďte následující kroky:

1. Projekt mějte v lokálním adresáři s právy pro čtení, zápis a spuštění.
2. Nainstalujte *js* závislosti pomocí

   ```
   npm install
   ```

   z kořenové složky a Python závislosti pomocí

```
pip install -r requirements.txt
```

ve složce *backend/*.

3. Sestaňte statické soubory pomocí:

```
npm run build
```

Nebo:

```
npm run build-dev
```

Soubory budou uloženy ve složce *dist/*.

4. Spusťte MongoDB server na defaultním portu.
5. Spusťte Flask server pomocí `python ajax_handler.py` ve složce *backend/*.
6. Spusťte webový server (např. Nginx) a nastavte cestu k statickým souborům na složku *dist/*. Ujistěte se, že server zároveň pomocí reverse proxy posílá požadavky na Flask server (port 5000).
7. Otevřete webový prohlížeč a zadejte adresu serveru.

5. Struktura projektu

Kořenová složka

- *audio/* — Obsahuje zvukové soubory pro efekty a hudbu.
- *backend/* — Skripty Flask aplikace a další backendové soubory.
- *css/* — Styly pro frontend.
- *dist/* — Výsledné soubory po sestavení.
- *fonts/* — Fonty použité ve hře.
- *img/* — Obrázky použité ve hře.
- *js/* — JavaScriptovské soubory pro frontend.
    - *js/constants/* — Konstanty použité ve hře.
    - *js/graphic_elems/* — Grafické elementy použité ve hře.
- *node_modules/* — NPM závislosti.

- *package.json* — Konfigurace pro NPM.
- *index.html* — Vstupní bod pro hru.

Backend (složka *backend/*)

- `db.json`: JSON soubor pro inicializaci databáze pro správný běh hry při prvním spuštění.
- `db_init.py`: Pythonovský skript pro inicializaci databáze.
- `ajax_handler.py`: Pythonovský skript pro zpracování AJAX požadavků s využitím Flask.
- `requirements.txt`: Závislosti pro Python.

Frontend (složka *js/*)

> Soubory jsou řazeny podle jejich přednosti ve spouštění. Jsou ukázány pouze hlavní soubory, ze kterých je evidentní struktura and fungování hry. Podrobnější popis jednotlivých metod a tříd je uveden v dokumentaci kódu či v souboru **jsdoc.html**. Všechny metody jsou podrobně zdokumentovány pomocí JSDoc.

- **app.js**: Vstupní bod pro hru.
    - Zde dochází k inicializaci hry.
    - Dojde k vytvoření instance PIXI aplikace.
    - Dojde k napojení metod zajistujících škálování a přizpůsobení velikosti okna a pro ovládání klávesnicí.
    - Dojde k načtení všech potřebných assetů (obrázky, zvuky, ...).
    - Vznikne instance hry, kterou reprezentuje třída `Game` z modulu `game.js`.
    - K metodě `update(delta)` z modulu **game.js** je připojen ticker poskytovaný PIXI.js, který zajišťuje pravidelné volání metody `update(delta)` s aktuálním časovým rozdílem mezi jednotlivými vykreslenými snímky.
        - Metoda `update(delta)` zajišťuje aktualizaci stavu hry a jedná se o hlavní smyčku hry.
- **loader.js**: Modul, který zajišťuje načítání assetů.
- **sound_manager.js**: Modul, který zajišťuje používání zvuků ve hře.
    - Obsahuje třídu `SoundManager`, která si drží informace o zvucích ve hře.
    - Každý zvuk se spoustí pomocí metody `playNázevZvuku()`.
    - Instanci třídy `SoundManager` používá třída `Game` a `GameSession` pro spouštění zvuků ve hře.
- **game.js**: Modul obsahující třídu `Game`, která reprezentuje hru.
    - Třída `Game` si drží informaci o stavu ve kterém se hra nachází pomocí proměnné `gameState` (jedná se o instanci třídy `GameState`, která je definována v modulu `game_state.js`).
    - Hlavní metodou je `update(delta)`, která zajišťuje aktualizaci stavu hry.
    - Metoda `update(delta)` je volána z `app.js` a zajišťuje pravidelné volání metody `update(delta)` s aktuálním časovým rozdílem mezi jednotlivými vykreslenými snímky.
    - Funguje na pricipu stavového automatu, kde se v závislosti na stavu hry provádí jiné aktualizace (pro rozhnodování byl použit switch case).
    - V závislosti na stavu hry se volají metody pro aktualizaci jednotlivých částí hry:
        - `handleMainMenuUpdate()`: Aktualizace hlavního menu na základě stisknutých kláves a vybraných možností.
        - `handleGameSessionUpdate(delta)`: Inicializace/aktualizace probíhající herní session.

        - `handleSettingsUpdate()`: Aktualizace nastavení hry na základě stisknutých kláves a vybraných možností.
        - `handleLeaderboardUpdate()`: Aktualizace pohledu žebříčku nejlepších hráčů.
        - `handleGameEndScreen()`: Aktualizace obrazovky po skončení hry.
    - Ve všech těchto metodách se pracuje se dvěma hlavními objekty:
        - `screenContent`: Obsahuje všechny logické objekty, které se mají vykreslit na obrazovku (v případě menu tedy jednotlivé možnosti, v případě herní session tedy herní pole, hráče, nepřátele, ...).
        - `drawingManager`: Jedná se jednu z instancí modulu `drawing_manager_menus.js` pokud se jedná o vykreslování určitého menu, nebo `game_session.js` pokud se jedná o vykreslování herní session.
            - Pokud se jedná o menu, tak se přímo volají metody pro vykreslení celého pohledu na základě aktualizace uživatelského vstupu. Překreslování se děje vždy, když je změněn stav menu nebo z důvodu změny velikosti okna.
            - Pokud se jedná o herní session, tak se neustále volá metoda pro aktualizaci herní session, která si sama spravuje vykreslování herního pole, hráče, nepřátel, ... a zajišťuje interakci mezi nimi včetně kolizí a zpracování vstupu od uživatele. Překresoloání se děje na úrovni herních objektů, které jsou vytvořeny pomocí PIXI.js nikoli na úrovni celého pohledu (z důvodu optimalizace a svižnosti překreslování).

            > Autorova původní myšlenka byla, že by se instance z modulu drawing_manager používaly pouze pro vykreslování na obrazovku, ale nakonec se ukázalo, že v případě herní session (kvůli složitosti implementace detekce kolizí bez přístupu k hernímu kanvasu atd.) bylo rozumnější nechat si spravovat vykreslování a interakce přímo herní session, neboť detekce a pohyb byl úzce spjat PIXI objekty a bylo by nepřehledné posílat aktualizace mezi různými moduly.

            - JavaScript sice nativně nepodporuje rozhraní, ale na základě autorem vytvořené struktury se očekává, že instance proměnné `drawingManager` bude vždy obsahovat metody:
                - `redraw()` pro překreslení herního pole po změně velikosti okna.
                - `cleanUp()` pro vyčištění kreslící plochy po přechodu na jiný pohled.
            - Všechny stavy zobrazující menu používají metodu `draw()`, která vykresluje `screenContent` na obrazovku.
            - Stav herní session používá metodu `update(delta)`, která zajišťuje aktualizaci herní session a v případě potřeby i překreslení herního pole.

- **game_session.js**: Modul obsahující třídu `GameSession`, která reprezentuje herní session.
    - Třída `GameSession` si drží informaci o stavu ve kterém se herní session nachází pomocí proměnné `gameSessionState` (jedná se o instanci třídy `GameSessionState`, která je definována v modulu `game_session_state.js`).
    - Jako první je nutné zavolat metodu `start()` pro inicializaci herní session.
    - Hlavní metodou je `update(delta)`, která zajišťuje aktualizaci stavu herní session.
    - Metoda `update(delta)` je volána z **game.js** a zajišťuje pravidelné překreslování a aktualizace herní logiky s aktuálním časovým rozdílem mezi jednotlivými vykreslenými snímky.

    - Funguje na pricipu stavového automatu, kde se v závislosti na stavu herní session provádí jiné aktualizace (pro rozhnodování byl použit switch case).
    - V závislosti na stavu herní session se volají metody pro aktualizaci jednotlivých částí hry:
        - `handleGameSessionInProgressUpdate(delta)`: Aktualizace probíhající herní session. Posloupnost aktualizací je následující:
            1. Aktualizace času hry.
            2. Aktualizace HUD pomocí metody `updateStats(delta)`.
            3. Aktualizace herních entit pomocí metody `updateEntities(delta)`. Tato metoda vrací informaci o případném zisku skóre, které je potřeba zpracovat.
                1. Nejprve se získají informace o všech entitách na herním poli (hráč, nepřátelé, bomby, bonusy, ...).
                2. Poté se aktualizuje entita hráče (kolize s entitami a poté pohyb).
                3. Poté se aktualizují nepřátelé (kolize s entitami a poté pohyb).
                4. Poté se aktualizují ničitelné zdi (kolize s entitami).
                5. Poté se aktualizují bomby (pokládání bomb, časovač výbuchu).
                6. Poté se aktualizují výbuchy bomb (vytvoření explozí, časovač trvání explozí).
                7. Nakonec se kontroluje, zda se můžou objevit únikové dveře (pokud jsou splněny podmínky).

            - > K aktualizacím dochází pouze pokud mometálně nedochází ke škálování herního pole (kvůli změně velikosti okna).

        - `handleGameSessionPlayerHitUpdate(delta)`: Aktualizace herní session po zásahu hráče nepřítelem.
            - Zajišťuje problikávání hráče při zásahu nepřítelem a aktualizaci životů hráče.
            - Pokud hráč ztratí všechny životy, tak se stav herní session změní na `GAME_SESSION_STATE_GAME_END`.
            - Pokud hráč má ještě životy, tak se stav herní session změní na `GAME_SESSION_STATE_LEVEL_INFO_SCREEN`.
        - `handleGameSessionLevelInfoScreen(delta)`: Aktualizace obrazovky s informacemi o levelu.
        - `handleGameSessionPausedUpdate()`: Aktualizace obrazovky po pozastavení hry.
        - `handleGameSessionLeavePromptUpdate(delta)`: Aktualizace obrazovky s výzvou k opuštění hry.
        - `handleGameSessionGameEndUpdate(delta)`: Metoda zajišťující správné ukončení herní session.
        - Většina těchto metod pracuje s proměnnou `screenContent`, která obsahuje všechny logické objekty, které se mají vykreslit na obrazovku (v případě herní session tedy herní pole, hráče, nepřátele, HUD, prompty, ...).

- **arena.js**: Modul obsahující třídu `Arena`, která reprezentuje herní pole.

    - Třída `Arena` si drží informace o herním poli (reprezentované 2D polem) vzhledem k obrazovce.
    - Obsahuje metody pro vykreslení a překreslení herního pole.
    - Obsahuje metody pro získání informací o herním poli (získání 2D indexu pole na základě souřadnic, získání souřadnic pole na základě 2D indexu, ...).
    - Obsahuje metodu pro ověření kolize s herním polem (metoda pro výpočet je předávána jako parametr).

    - Také obsahuje metody
        - `draw()`: Metoda pro vykreslení herního pole.
        - `redraw()`: Metoda pro překreslení herního pole.
        - `cleanUp()`: Metoda pro vyčištění herního pole.

- **entity.js**: Modul obsahující třídu `Entity`, která reprezentuje herní entitu.

    - Třída `Entity` si drží informace o herní entitě vzhledem k hernímu poli.
    - Obsahuje metody pro vykreslení entity při jejím vytvoření a aktualizaci entity (detekce kolize a pohyb).
    - Všechny herní entity (hráč, nepřátelé, bomby, exploze, ...) dědí od třídy `Entity` a přepisují metodu `update(delta)` na základě svých potřeb.
    - Obsahuje metody:
        - `spawn(x, y)`: Metoda pro vytvoření entity na herním poli na základě souřadnic.
        - `update(delta)`: Metoda pro aktualizaci entity.
        - `redraw()`: Metoda pro překreslení entity (při změně velikosti okna).
        - `moveToTop()`: Metoda pro přesunutí entity do popředí.
        - `remove()`: Metoda pro odstranění entity z herního pole.

- **levels_config.js**: Modul obsahující konfiguraci levelů pro *normal* mód.

> Většina elementů je pozicována/škálována na základě předem specifikovaných poměrových konstant (vzhledem k šířce/výšce obrazovky/určité entity). Všechny tyto konstanty jsou buď definovány na začátku modulu, na začátku třídy nebo jsou předávány jako parametry metodám.

6. Testování

Aplikace byla testována pouze pomocí manuálních testů. Testování bylo zaměřeno na ověření funkčnosti hry a správného chování herní logiky. V rámci projektu proběhlo také testování od 3 nezávislých testerů, kteří měli za úkol ověřit funkčnost hry a zpětně poskytnout zpětnou vazbu, která byla následně zohledněna při vývoji. Zpětná vazba od testerů:

- Kateřina Hanělová (*současná studentka předmětu KIV/UUR*)

    - Odezva: *„Aplikace je jednoduše a pěkně graficky zpracovaná, vše je na obrazovce rozloženo intuitivně. Doplnění vizuálu různými trefnými zvuky mě příjemně potěšilo. Ovládání menu, nastavení i hry je jednoduché a intuitivní. Líbí se mi možnost nastavení počtu životů a také módu hry, aby si uživatel přizpůsobil obtížnost podle své chuti. Samotná hra je jednoduchá a zabaví hráče na delší dobu, díky tomu, že je stále zábavná a levely jsou různé a postupně náročnější. Také udrží díky hezkému grafickému provedení. Při testování jsem na chyby nenarazila, vše fungovalo tak, jak jsem očekávala.“*
    - Autorova reakce: Potěšení z pozitivní zpětné vazby. Díky pozitivní zpětné vazbě ohledně intuitivního ovládání se autor zaměřil na tento aspekt a při testování odhalil, že se hra nedá pozastavit v herní session při aktivním CAPS LOCKu (hra očekávala písmeno *'p'*, ale CAPS LOCK přepisoval klávesu na *'P'*). Tento problém byl následně opraven. Dále byla nalevo od herního plátna přidána jednoduchá legenda pro klávesové ovládání.

- Jiří Winter (*absolvent předmětu KIV/UUR*)
    - Odezva: *„Hra Bomberman Clone je pro mě velice zajímavé téma semestrální práce a testování bylo velice jednoduché a záživné. Hra i menu běží v pořádku a i při intenzivnějších pokusech o způsobení chyby, běžela v pořádku. Hra je responzivní a jednoduchá na ovládání. Na první dobrou jsem pochopil, jak ji ovládat a tutorialu nebylo vůbec potřeba. Líbí se mi, že byla zvolena stoupající obtížnost levelů od velice jednoduché po náročnější, kde jsem musel chvilku zůstat, abych se levelem probojoval. Jedinou výtku mám k menu a nemožnosti použití myši. Sice tomu rozhodnutí rozumím, aby bylo patrné, že ovládání je jen klávesy, ale jelikož se jedná o webovou aplikaci, uvítal bych možnost ovládání myší v menu. I přesto si ale myslím, že aplikace je velice povedená a hraní mi bavilo.“*
    - Reakce autora: Důvod absence myši byl záměrný. Hra je stylizována jako stará arkáda. Myš by mohla způsobit zmatení uživatele, kdyby ji mohl použít v menu, ale ne ve hře.
- Luboš Hess (*student přírodních věd, volnočasový hráč PC her*)
    - Odezva: *„Rychlost pohybu hráče se zdá být příliš pomalá. Ze začátku mě kvůli tomu trochu štvalo zasekávání se o hrany nezničitelných zdí, ale po čase jsem si zvykl. Často se mi stávalo, že jsem se objevil v novém levelu a byl jsem okamžitě obklopen nepříteli a neměl jsem možnost úniku. Jakmile nepřátelé dorazí k bombě, zastaví se u ní a vyčkávají dokud nevybouchne. Když jsem si nastavil hru na 1 život, po 1. zásahu hra skončila, ačkoliv měla dále pokračovat. Ve 4. levelu normal modu je nepřítel umístěn doprostřed mapy mezi bloky. Po odstranění bloků se záporák začne pohybovat a okamžitě umírá na následky výbuchu položené bomby. V endless modu se v každém levelu objeví zničitelná zeď v levém horním rohu. Bylo často zdlouhavé se neustale dostávat k tomuto bloku. V menu nastavení bych uvítal možnost ovládat volby pohybem kláves doleva a doprava. Je to standardní v mnoha starých i nových hrách.“*
    - Autorova reakce: Rychlost pohybu byla nastavena na danou hodnotu záměrně, jelikož je hra velmi elementární, tak část její výzvy tvoří vyhýbání se nepřátelům. Pokud by byla rychlost pohybu vyšší, tak by se hra stala příliš snadnou nebo pro nezkušené hráče by vytvořila jednoduší způsob jak naběhnout do nepřátel. Při zvýšení rychlosti všech entit se poté hra stala příliš náročnou, kvůli zkrácenému reakčnímu času na vyhýbání se nepřátelům. Situace s „insta-killem“ byla způsobena příliš malým radiusem okolo hráče, kde se neměly při generování levelu objevit žádné nepřátelské entity. Tento problém byl opraven. Vlastnost nepřátel, že chodí do bomb je úmyslná. Bystří hráči by mohli být schopni si vypočítat trajektorii pohybu nepřítele a umístit mu do cesty bombu. Chyba s počtem životů byla způsobena špatným nastavením podmínek pro ukončení hry. Tento problém byl také opraven. Případ sebedestruktivních nepřátel byl vyřešen vylepšením jejich AI na hledání cest k hráči. Nezničitelná zeď v levém horním rohu byla způsobena ladící entitou, která byla zapomenuta v kódu. Tato zničitelná zeď se již pravidelně negeneruje. Možnost ovládání menu klávesnicí byla přidána.

7. Algoritmické a implementační detaily

Způsob kreslení herních elementů v game session prošel v průběhu vývoje velkými změnami. V původní verzi bylo vykreslování a pohybování entit řešeno zvenčí. V průběhu vývoje se ukázalo, že je lepší nechat si spravovat vykreslování a pohybování entit přímo v jejich vlastních instancích, neboť jednotlivé odchylky způsobů aktualizací, založené na typu entity, se daly řešit děděním od nadřazeného, implicitního objektu `Entity`. Před posláním verze testerům tedy došlo k větším změnám v kódu.

Jinak je kód dle autorova úsudku napsaný se standardními postupy pokud se jedná o problematiku jako jsou stavové automaty, zobrazování grafických primitiv na plátně, preškálování, řešení hit testů, pohyb na základě delta času, atd. Úsek kódu, který stojí za zmínku, neboť se jedná o vlastní verzi algoritmu je pohyb nepřátel z modulu **enemy.js**. Způsob jakým se nepřítel pohybuje je následující:

```
1. Nepřítel si zkontroluje, zda má nastavený cíl.
2. Pokud nemá cíl, tak se pokusí najít cíl následujícím způsobem:
   1. S určitou pravděpodobností (na základě obtížnosti nepřítele) si
vybere pokud jeho cíl bude hráč nebo náhodný bod na herním poli.
   2. S určitou pravděpodobností (na základě obtížnosti nepřítele) si
vybere jakým způsobem vypočítá cestu k cíli.
       - Při vyšší obtížnosti se spíše použije BFS algoritmus pro nalezení
nejkratší cesty k cíli.
       - Při nižší obtížnosti se spíše použije DFS algoritmus pro nalezení
nejkratší cesty k cíli (varianta s náhodným výběrem větve).
   3. Vypočítaná cesta k cíly je tvořena seznamem 2D indexů herního pole.
3. Pokud má cíl, tak se pokouší jednoduchým přičítáním a odečítáním od
souřadnic dostat na následující bod v cestě k cíli (vzdálenost je
vypočítána v souladu s rychlostí hráče)
4. Takto se pokouší dostat na cíl, dokud není v dosahu cíle.
5. Pokud je v dosahu cíle, tak se cesta k cíli zahodí a algoritmus se
opakuje od bodu 1.
```

8. Závěr

Aplikace byla úspěšně dokončena a splnila všechny povinné požadavky ze zadání. Během vývoje projektu si autor rozšířil znalosti v oblasti JavaScriptu, práce s knihovnou PIXI.js a backendového vývoje v Pythonu s Flaskem. Hra byla otestována třemi nezávislými testery a získaná zpětná vazba byla zohledněna při dalším vývoji, což vedlo k vylepšení uživatelského rozhraní a herní logiky.

Autor reflektuje na tento projekt jako na významný krok ve svém studiu, který mu umožnil propojit teoretické znalosti s praktickou realizací software. Cenné bylo zejména řešení reálných programátorských výzev, jako je optimalizace výkonu hry a zajištění plynulého uživatelského zážitku.

Aplikace by dále mohla být rozvíjena přidáním dalších herních módu, vylepšením AI nepřátel pro zvýšení obtížnosti, přidáním power-upů a bonusů a implementací mobilní verze hry, aby byla přístupná širšímu spektru hráčů. Tento projekt posílil autorovy schopnosti jako vývojáře software a ukázal důležitost základů uživatelského rozhraní a herní logiky pro vytvoření kvalitního produktu.