Anna Vičíková

Dokumentace k webové hře Snake

Semestrální práce

Anna Vičíková
24.5.2020

Obsah

1 Cíl projektu

Cílem tohoto projektu je vytvořit webovou aplikaci, která bude obsahovat známou hru Snake. Při spuštění hry se objeví had, kterého hráč bude moct ovládat pomocí pomocí šipek na klávesnici. Cílem hry je sníst co nejvíce jablek, za každé snězené jablko se přičítá bod. Hra skončí, pokud had narazí do svého těla. Na konci hry se objeví finální skóre a zapíše se do žebříčku nejlepších hráčů.

2 Vývojové prostředí a využité prostředky

Hra byla vytvořena v editoru Visual Studio Code od společnosti Microsoft.

Byla využito

- Windows web softwarový balíček WampServer.
    - Apache, MySQL, PHP / phpMyAdmin
- Platforma Firebase od společnosti Google

3 Popis implementace

Pro vytvoření této hry bude využit HTML5 Canvas element. Na počátku se vytvoří základní index.html s canvas elementem o výšce a šířce 500x500, k němu kaskádové styly. Funkčnost je rozdělena do následujících částí:

> ***main.js*** *– hlavní javascriptový soubor, zavádí modální okna, spuštění, běh a konec hry. Později obsahuje i konfiguraci pro Firebase platformu.*
>
> ***Snake.js*** *– třída popisující hada. Je zde implementována veškerá logika hada, včetně jeho pohybu, načítání bodů (zvětšování těla, snězení jablka) a detekování kolize.*
>
> ***Apple.js*** *– třída popisující jablko. Nasazuje souřadnice jablka a jeho vykreslení.*
>
> ***Sound.js*** *– třída sloužící pro nastavení zvuku.*

3.1 Základní hra

Cílem této části bude vytvořit pouhou funkční hru.

3.1.1 Ovládání

Hra se bude ovládat pomocí šipek. Definujeme si tedy v souboru main.js možné stavy v proměnné „*Direction“*, která bude zaznamenávat směr stisknuté šipky na klávesnici. Nastavíme si listener na event *„keydown“* a podle stisknuté klávesy se nastaví směr hada. Nastavení směru u hada bude ovládat metoda *setDirection* ve třídě **snake.js**, který si přebírá směr do vlastní proměnné *next*, která určuje, do jakého směru se had vydá. Kromě toho ještě využijeme *preventDefault()* metody, aby šipky neovládaly scrollování stránky, pokud k tomu nastane.

Ve třídě **Snake** si nadefinujeme konstruktor, kde si nadefinujeme aktuální směr, příští směr souřadnice hlavy a pole, které bude sloužit jako tělo hada a metody *setDirection* a *update*. *Update* metoda bude sloužit jako hlavní část, která se bude vždy volat v cyklu. V ní si nadefinujeme, aby další směr, který bude nastaven, se nemohl přepnout do směru opačného, tzn. že by had poté přejel sám přes sebe do opačného směru. Aktuální směr se ukládá do proměnné *movement*. Dále v metodě update si nadefinujeme přepínač pro směr, který bude po určitém směru přičítat (resp. odečítat) souřadnice hada o jeden blok.

3.1.2 Trocha matematiky

Herní plocha je o rozměrech 500x500. Určíme si velikost bloku hada jako 25x25 pixelů. To znamená, že herní plocha bude rozdělena na 20 bloků.

3.1.3 Vykreslení hada

Vytvoříme si metodu *draw(ctx)*. Za každý blok v těle se v canvas vykreslí čtvereček o rozměrech 23x23 pixelů (z kosmetických důvodů) a vykreslíme si hlavu zvlášť.

3.1.4 Pohyb tělem hada

Pro pohyb budeme jednoduše přidávat souřadnice na konec pole, které symbolizuje tělo hada, a odebrání první souřadnice tohoto pole. Tím tak simulujeme skutečný pohyb hada.

3.1.5 Hranice herní plochy

V této hře se nebude prohrávat nad tím, že had narazí do hranice herní plochy. Když had projede přes hranice, pokračuje na protější straně hranice.

3.1.6 Jablko nepadá daleko od stromu.

V **Apple.js** si vytvoříme třídu Apple se souřadnicemi a vykreslovací metodou. Jablko bude vykreslováno jako bílý puntík.

Nyní si naimplementujeme kolizi hada s jablkem v metodě *update* v **Snake.js**. Pokud bude mít jablko stejné souřadnice jako hlava hada, nastane kolize, tzn. had jablko sní. Přičte se skóre a prodlouží se tělo hada a jablko se vygeneruje na novém místě.

3.1.7 Herní cyklus

v **main.js** si nadefinujeme metodu *startGame()*. Po definování hada a jablka si nadefinujeme zde herní cyklus pomocí metody *setInterval*, ve které budeme volat u hada metodu *update* a u hada a jablka metodu *draw*. Do teď jsme naimplementovali úplný základ logiky hry.

3.1.8 Konec hry

V **Snake.js** si naimplementujeme metodu *gameOver()*, která nám detekuje, jestli hlava hada nenarazila do svého těla. Pokud se to tak stalo, v main.js si pak zavoláme metodu, která nám vykreslí Game Over okno, kterou si naimplementujeme později.

3.2 Skóre

Chtěli bychom za každé snězení jablka si přičíst skóre. Nyní si přičítáme skóre pouze v proměnné náležící hadovi, chtěli bychom ho vykreslit do stránky. Definujeme si tedy HTML element *span*, kterému přiřadíme id, tento objekt si podle jeho id uložíme do proměnné v konstruktoru hada. Při každém snězení jablka se přičte bod do proměnné *score*, poté se toto *score* překreslí v uloženém *spanu*.

3.3 Modální okna

Důležitou část v této semestrální práci hrají roli modální okna. Naimplementujeme si zde v main.js dvě okna – hlavní menu a poté Game over okno, kde se vyplní jméno hráče.

3.3.1 Hlavní menu

Pro vykreslování menu využijeme template literály. V metodě *createMainMenu()* si vykreslíme formulář, kde si hráč zvolí obtížnost (ta bude naimplementována později) pomocí inputu typu „*radio*“ a bude zde tlačítko na spuštění hry. Na tomto tlačítku si nastavíme listener na stisknutí tohoto tlačítka, která způsobí, že se menu skryje a spustí hru.

3.3.2 Game Over okno

Po tom, co nastane kolize hlavy hada se svým tělem, nastane konec hry a vykreslí se Game Over okno. V metodě *createGameOver* si opět definujeme html elementy pomocí template literálů do prvku *modal*, který se tím pádem překreslí z původního hlavního menu. Vykreslíme si tedy formulář, kde bude textové pole pro zadání jména a tlačítko na odeslání tohoto jména. Ošetříme, aby hráč neodeslal prázdné textové pole, ale taktéž aby textové pole nebylo příliš dlouhé (kvůli výpisu do žebříčku hráčů). Po kliknutí na tlačítko odeslání se přesuneme na žebříček, který naimplementujeme hned v dalším kroku.

3.4 Žebříček nejlepších

Chtěli bychom vidět pořadí hráčů, u kterých uvidíme jméno a jejich skóre, seřazených samozřejmě od nejlepšího (tzn. od nejvíce bodů po nejméně bodů). V metodě *createLeaderboard()* si okno žebříčku vytvoříme.

3.4.1 Ukládání skóre do Local Storage

Jedna z variant, jak si ukládat skóre, je přes Local Storage Web API. Na začátku main.js si nadefinujeme *storage*. Při definování Game Over si jméno a hráčovo skóre uložíme do tohoto *storage*, naimplementujeme si metodu, která tento *storage* uspořádá podle skóre a v žebříčku si tyto data vyčteme.

3.4.2 Ukládání skóre do databáze + Firebase

Z důvodu kompetetivnosti s dalšími hráči, kteří nesdílí jeden prohlížeč, jsem se rozhodla, že využiji databázi. Našla jsem platformu Firebase, který poskytuje vlastní backend s databází a tak jsem se této příležitosti uchopila.

O tom jak zprovoznit Firebase na svém projektu, jsem se řídila dokumentací této platformy na adrese

https://firebase.google.com/docs/web/setup.

Celý proces taktéž zahrnul přidání SDK do index.html a konfigurace do main.js. Následně se musí nastavit i databáze přímo na této platformě.

V main.js si můžeme referenci na databázi uložit do *scoresDb*, kde přímo čteme tzv. „dokumenty" s jejich hodnotami. V této práci jsem jako „dokumenty" vytvářela po každém jméně hráče, které bylo zadáno a pak se k nim přiřadila hodnota skóre. V metodě *createLeaderboard* přečteme všechny data a uložíme si ji do vlastní mapy, kterou opět uspořádáme podle skóre, pomoci promise.

3.5 Obtížnost

Aby hra nebyla monotónní, přidáme do hry obtížnost. V hlavním menu jsme si obtížnost definovali již dříve, nyní toho využijeme. Obtížnost hry bude následující:

Easy – nejlehčí – skóre se bude přičítat po jednom bodu a tělo hada po jednom bloku. Rychlost hada (obnovení hada každých několika milisekund pomocí setInterval metody) je nastavena na 120 ms.

Medium – střední – skóre se bude přičítat po dvou bodech a tělo hada po dvou blocích. Rychlost hada je nastavena na 80 ms.

Hard – těžká – skóre se bude přičítat po pěti bodech a tělo hada po pěti blocích. Rychlost hada je nastavena na 50 ms.

Podle toho, jakou obtížnost si zvolí hráč, si tuto možnost předáme přímo do hry. Rychlost hada nastavíme v metodě *setInterval*, podle obtížnosti se do přičítání skóre přidá určitý počet bodů (přiřazený té obtížnosti) a podle toho taktéž se budou přidávat určitý počet bloků do těla hada.

3.6 Hudba a zvuky

Do tohoto projektu byly nahrány určitá hudba a zvuky. Jsou zde tři zvuky – soundtrack, který běží celý herní cyklus taky v cyklu, zvuk, když had sní jablko a poté zvuk, když nastane konec hry.

Pro implementaci hudby si vytvoříme třídu *Sound* v **Sound.js**, kde si v konstruktoru zvuk nastavíme.

Při spuštění zvuku u sněženého jablka, tedy získání bodu, dojde k tomu, že tento zvuk je stále aktivní a pokud hráč sní další jablko, tak se zvuk opět nezapne. Tohle řeší klonování zvuku, pokud původní zvuk ještě stále nebyl dohrán.

3.7 Offline aplikace

Aplikace je podporována i offline díky Service Worker API. V index.html se vypíše script, který hlídá registraci service wrokeru. Service worker je implementován v sw.js. Do cache se nahrají soubory, které se poté řídí fetch eventem.

Výhoda, že aplikace využívá Firebase, je, že když je klient offline, zahraje hru, zadá své jméno a odešle skóre do žebříčku, tak jakmile se znovu připojí k internetu, tak se požadavek na přidání skóre do Firebase databáze spracuje.

3.8 Aplikace na menším okně

Protože hra by na malém okně nebyla hratelná, je zde implementováno Media query, které skryje všechny herní prvky a ukáže pouze žebříček.

4 Příručka uživatele

Při načtení stránky se Vám zobrazí hra. Při stisknutí spodního tlačítka „Leaderboard“ si můžete načíst žebříček hráčů a jejich skóre. Při stisknutí tlačítka „Game“ se můžete vrátit zpátky na hlavní obrazovku hry. Vyberte požadovanou obtížnost. Čím vyšší obtížnost, tím rychlejší had bude a tím delší bude po snězení jablka.

Hra se ovládá šipkami na klávesnici. Při hraní hry nemůžete přejít do žebříčku, takovou možnost budete mít až po skončení hry.

Po skončení vyplňte své jméno a odešlete svůj výsledek. Budete přesměrováni na žebříček.