Projektová dokumentace

Š - ARM-FITkit3 či jiný HW: Hra HAD

Jan Klanica - xklani00

1. Úvod

Tato projektová dokumentace se zabývá popisem problematiky, způsobu implementace a popisem dosažených výsledků v rámci projektu Hra HAD na ARM-FITkit3 do předmětu IMP.

2. Hra Had

Hra Had je klasická počítačová hra, ve které hráč ovládá hada, jenž se pohybuje po obrazovce. Cílem hry je sbírat jablka nebo jiné objekty, které se objevují na obrazovce, a tím zvětšovat délku hada. Každý nový objekt, který had sní, způsobí, že se had prodlouží, což ztěžuje hru, protože had se stává delší a hůře se vyhýbá překážkám, včetně sebe sama. Hra končí, pokud had narazí do stěny nebo do své vlastní části těla.

V některých verzích hry však stěny nejsou přítomny, a místo toho hra používá princip "wrap-around" nebo "tiled world", kdy had pokračuje na opačnou stranu obrazovky, pokud narazí do okraje. Tento způsob zjednodušuje hru, protože se hráč nemusí obávat neustálého vyhýbání se stěnám. V těchto verzích hra končí, pokud had narazí do své vlastní části těla.

Hra Had je velmi populární. Byla součástí operačního systému Nokia v 90. letech, a od té doby se stala symbolem jednoduchých, ale návykových her.

3. Implementace na ARM-FITkit3

3.1. Display

Hra Had se obvykle hraje na maticové ploše s velkými rozeznatelnými body, kde jeden bod představuje pozici, na které může v jednu chvíli být pouze tělo hada, jablko nebo prázdno. Z tohoto důvodu je k ARM-FITkit3 rozdáván ještě zobrazovací LED modul s rozměry 16x8 pixelů.

S ohlédnutím na to, že je pixel jednobarevný (svítí / nesvítí), rozhodl jsem se zobrazit hru Had na tento display takto:

- Jeden pixel na display představuje jeden bod na maticové ploše hry had.
- Bod, kde je tělo hada, je zobrazen jako svítící LED dioda.
- Bod, kde je jablko, je zobrazen jako svítící LED dioda.
- Bod, kde je prázdno, je zobrazen jako nesvítící LED dioda.

Jsem si vědom toho, že bod těla hada a bod jablka jsou na display nerozeznatelné. V rámci testování jsem ale zjistil, že si člověk většinou dokáže všimnout, kde se jablko zobrazilo a zapamatovat si tak jeho pozici.

Display je ovládán pomocí GPIO pinů portů A a E nastavených na výstup. Podrobnosti naleznete v dokumentaci pro ARM-FITkit3 a zobrazovací LED modul. Časování střídání sloupců je v sekci 3.3..

3.2. Uživatelské rozhraní

Had se na maticovém ploše může pohybovat do čtyřech směrů, respektive: nahoru, dolů, doleva, doprava. Pro tento případ jsem použil čtyři tlačítka dostupná na ARM-FITkit3, která svým umístěním vůči sobě tyto směry implicitně představují.

Je potřeba zmínit, že display se zdá býti oproti tlačítkům o 90 ° otočen. Pro chod hry to není žádný problém, protože kamera hry se na maticovou plochu dívá z pohledu ze shora a uživatel je schopen interpretovat směr tlačítek podle toho, jakým způsobem ARM-FITkit3 zrovna chytil. Pouze při implementaci bylo například určité tlačítko označeno jako směr nahoru ale pak převedeno do orientace displaye na směr doleva.

Tlačítka jsou čtena pomocí GPIO pinů na portu E nastavených na vstup. Konkrétně jsou využita tlačítka SW2, SW3, SW4, SW5. V případě stisknutí je vyvoláno přerušení, které se zpracuje. Pokud je zmáčknuto několik tlačítek po sobě v jednom kroku hry, použije se poslední zmáčknuté. Pokud je současně zmáčknuto více tlačítek, výsledný směr se odvíjí od priority níže (od nejvyšší priority po nejnižší):

- dolů
- nahoru
- doleva
- doprava

3.3. Časování chodu hry a displaye

Ve hře had se pravidelně každých N sekund posune had o jeden bod ve směru určeném uživatelem (nebo ve směru stejném jako předešly posun). Pro pravidelné posunovaní byl použit modul PIT (Periodic Interrup Timer), který dokáže vyvolat přerušení pravidelně po nějaké periodě.

Hodnota TSV pro registr PIT_LDVAL byla vypočítána následovně:

$$TSV = (T * f_{count}) - 1 \tag{1}$$

$$TSV = (0{,}2\ s * 50\ MHz) - 1\ =\ 9999999 \tag{2}$$

kde *T* je perioda posunu hada a f_{count} je frekcence PIT časovače.

Obodobnou úlohu je potřeba řešit i pro display, u kterého se dá v jednu chvíli ovládat pouze jeden sloupec. Rychlým střídáním sloupců se dá docílit rozsvícení celého displaye. Frekvenci displaye jsem si nastavil na 60 Hz. Výpočet níže:

$$f_{disp} = 60\ Hz \rightarrow\ f_{col} = 60\ Hz\ *\ 16\ = 960\ Hz \rightarrow\ T_{col}\ =\ \frac{1}{960} s \tag{3}$$

$$TSV = (\frac{1}{960}\ s * 50\ MHz) - 1\ =\ 52082 \tag{4}$$

3.4. Hra HAD

Maticovou plochu prostředí hry hada jsem reprezentoval jako dvorouzměrné pole obsahující hodnoty:

- ST_UP - je zde část hada; další část bližší k hlavě je směrem nahoru
- ST_LEFT - je zde část hada; další část bližší k hlavě je směrem doleva
- ST_DOWN - je zde část hada; další část bližší k hlavě je směrem dolů
- ST_RIGHT - je zde část hada; další část bližší k hlavě je směrem doprava
- ST_REWARD - je zde jablko
- ST_NONE - je zde prázdné místo

Rovněž jsem si pamatoval souřadnice prvního a posledního článku hada. Informace v buňce prvního článku hada má speciální význam. Udává poslední směr pohybu hada.

Algoritmus pro pohyb hada v určitém směru je vyjádřen pseudokódem níže:

```
FUNCTION snake_game_move(sgame, direction):
       // Získej aktuální směr hada
       current_direction = GET_CURRENT_DIRECTION(sgame)

       // Zamez pohybu proti směru
       IF direction == NONE OR direction == OPPOSITE(current_direction):
              direction = current_direction  // Zachovej stávající směr

       // Získej stav pro nový směr
       head_state = GET_STATE_FROM_DIRECTION(direction)

       // Vypočítej novou pozici hlavy
       new_head = CALCULATE_NEW_HEAD_POSITION(sgame, direction)

       IF new_head IS REWARD:  // Pokud had snědl odměnu
              PLACE_NEW_REWARD(sgame)  // Umísti novou odměnu
       ELSE:  // Pokud odměna není snědena
              MOVE_TAIL(sgame)  // Pohni ocasem, abys udržel délku hada

       IF COLLISION_WITH_SELF(sgame, new_head):  // Kontrola kolize se sebou samým
              RESET_GAME(sgame)  // Resetuj hru, pokud došlo ke kolizi

       // Aktualizuj pozici hlavy hada
       UPDATE_HEAD_POSITION(sgame, new_head, head_state)
```

4. Ukázka funkčnosti

https://nextcloud.fit.vutbr.cz/s/FSMjBE9TRob2gWa

5. Závěr

Můj projekt splňuje všechny požadavky a je plně funkční. Implementace úspěšně využívá maticový LED displej k zobrazování hada a jablka, s efektivním časováním kroků hry pomocí PIT časovače pro plynulý pohyb.

Celý projekt mi poskytl cenné zkušenosti s ARM-FITkit3 a programováním přerušení, práce s GPIO piny a časovači, což lze využít pro podobné aplikace na embedded zařízeních.

6. Autoevaluce

Skromně očekávám 14 bodů.