Fakulta aplikovaných věd

Katedra kybernetiky

Ping Pong

Semestrální práce z KKY/ZDO

Martin Skála

22.6. 2011

Řešitelský tým:

Martin Skála – ma.ska@post.cz

Zadání úlohy:

Vytvořit hru ve stylu ping pongu. Hra bude pro jednoho hráče, který bude snímán webkamerou a hru bude ovládat pohybem ruky, ve které bude mít předmět (např. propisku), který bude představovat pingpongovou pálku.

Specifikace úlohy:

Hra bude vytvořena pomocí nástroje Adobe Flex 4 a pomocí JavaScriptu. Uživatel bude snímán webkamerou. Obraz z webkamery bude pokud možno v reálném čase zpracováván a budou z něj získány informace o pohybu předmětu, kterým bude hra ovládána. Předmět, který bude uživatel držet v ruce, bude rovný, např. propiska a bude mít jiný jas než okolí (ruka uživatele, oblečení), aby se dal snadněji detekovat. Celá aplikace bude spustitelná v internetovém prohlížeči.

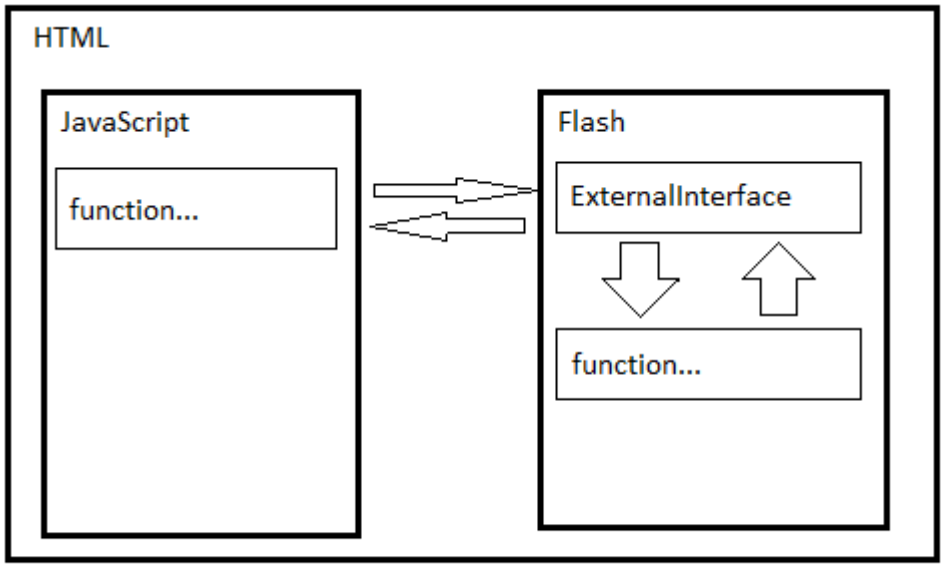
Řešení:

1) **Snímání obrazu**

   Snímání obrazu je zajištěno pomocí nástroje Adobe Flex 4 (Adobe Flash Builder 4), který je schopen ovládat webkameru a snímat z ní obraz, který můžeme dále zpracovávat. V programu je vytvořen objekt camera, který snímá obraz, ten se ukládá do BitmapData a následně do bytového pole, ve kterém je obraz v RGBA hodnotách. Pro jeden pixel jsou tedy v poli 4 hodnoty v celých číslech od 0 do 255.

2) **Odeslání dat jo JavaScriptu**

   Když máme data uložená v poli, přetypujeme ho na String a odesíláme do JavaSctriptu, kde probíhá další zpracování. K odesílání dat je použita třída ExternalInterface, která dokáže komunikovat s JavaScriptem. JavaScript tedy svojí funkcí zavolá přes ExternalInterface funkci, která je ve Flashy, ta získá data z webkamery a hodnoty vrátí opět přes ExternalInterface do JavaScriptu.

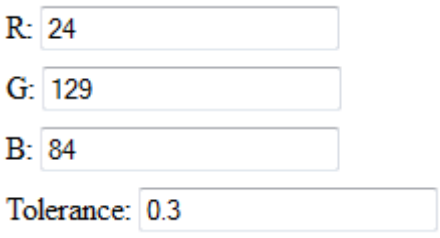


V našem případě probíhá snímání obrázku po určitých intervalech. Díky tomu, že odesíláme relativně velké množství dat (v našem případě máme obrázek 320x240 pixelů, tedy 320x240x4 hodnot) se program zpožďuje. To má za následek to, že intervaly snímání musí být větší a tím pádem se program znatelně seká. Pokud bychom odesílali méně dat, nemusela by zase detekce obrazu probíhat tak, jak bychom potřebovali.

3) Zpracování dat a detekce polohy objektu

Zpracování dat probíhá v JavaSctriptu, kde pomocí objektu canvas (který je novinkou v HTML5) můžeme vykreslovat obrázky a pracovat s jednotlivými pixely.

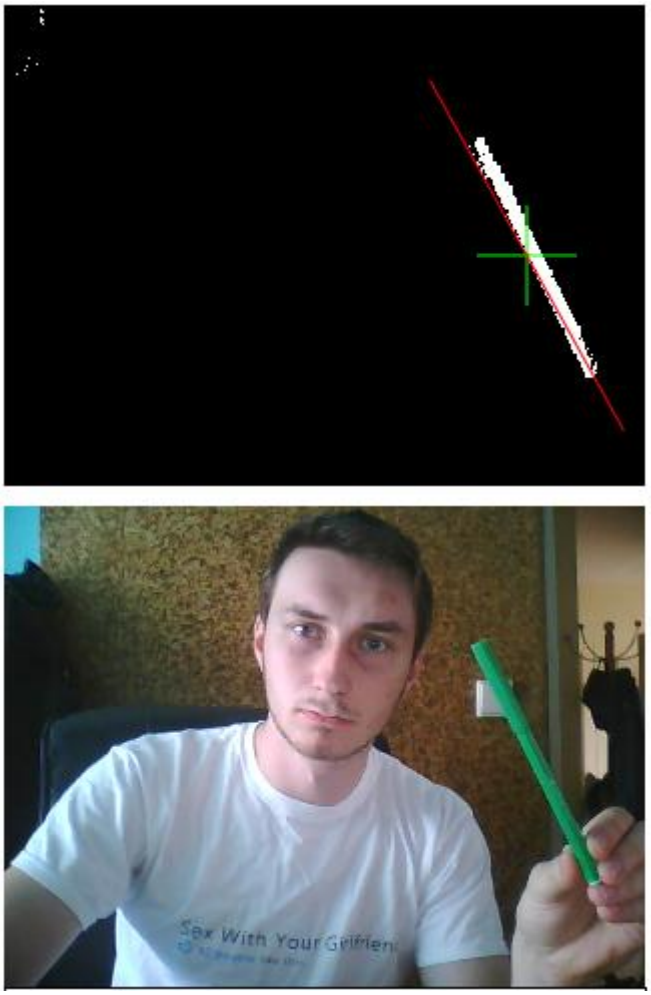
Nejprve si vždy musíme zvolit barvu detekovaného objektu a zadat ji do programu v RGB hodnotách. Barva objektu musí být odlišná od okolí, aby detekce mohla probíhat správně. Dále zvolíme toleranci barvy.



Tolerance nám vyjadřuje, jak moc se muže lišit barva obrázku od zadané barvy, abychom ji ještě detekovali jako stejnou. Hodnota 0.3 vyjadřuje 30% toleranci. Tolerance se počítá z RBG složek a je to součet rozdílu hodnot od zadané barvy. Pokud se tedy například hodnota červená liší o 5% od zadané, hodnota zelené se liší o 10%, tak při toleranci 30% se hodnota modré může lišit maximálně o 15%, abychom ji detekovali jako součást našeho objektu.

Tolerance se zadává z důvodu, že na různých částech objektu jsou jiné jasové hodnoty a tím pádem tedy různé RGB hodnoty.

Při detekci polohy objektu procházíme celý obrázek po pixelech a rozhodujeme na základě zadané barvy a tolerance, které pixely patří našemu objektu a které ne. Viz obrázek níže, kde jsme přebarvili vhodné pixely na bílo a nevhodné na černo.



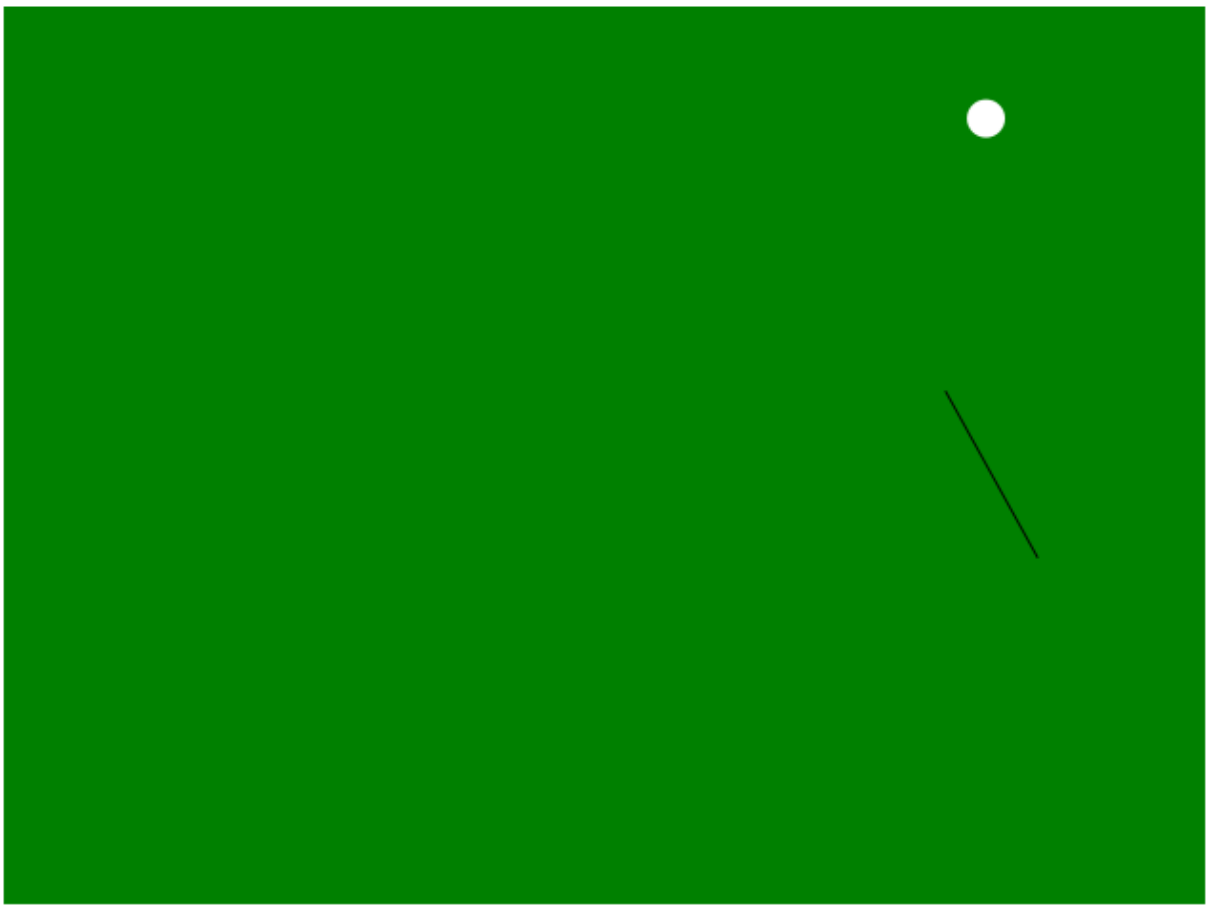
Na obrázku si můžeme všimnout dalších bílých pixelů, které nejsou součástí našeho objektu a přináší tak drobné zkreslení do výpočtu polohy objektu (pozadí má totiž podobnou barvu jako objekt a vejde se do tolerance).

Při výpočtu polohy objektu, proházíme obrázek a sčítáme x-ové hodnoty detekovaných pixelu a také y-onové. Pokud tyto dvě čísla vydělíme počtem detekovaným pixelů (uděláme průměr) dostaneme x a y hodnotu středu našeho objektu, viz zelený křížek na obrázku výše.

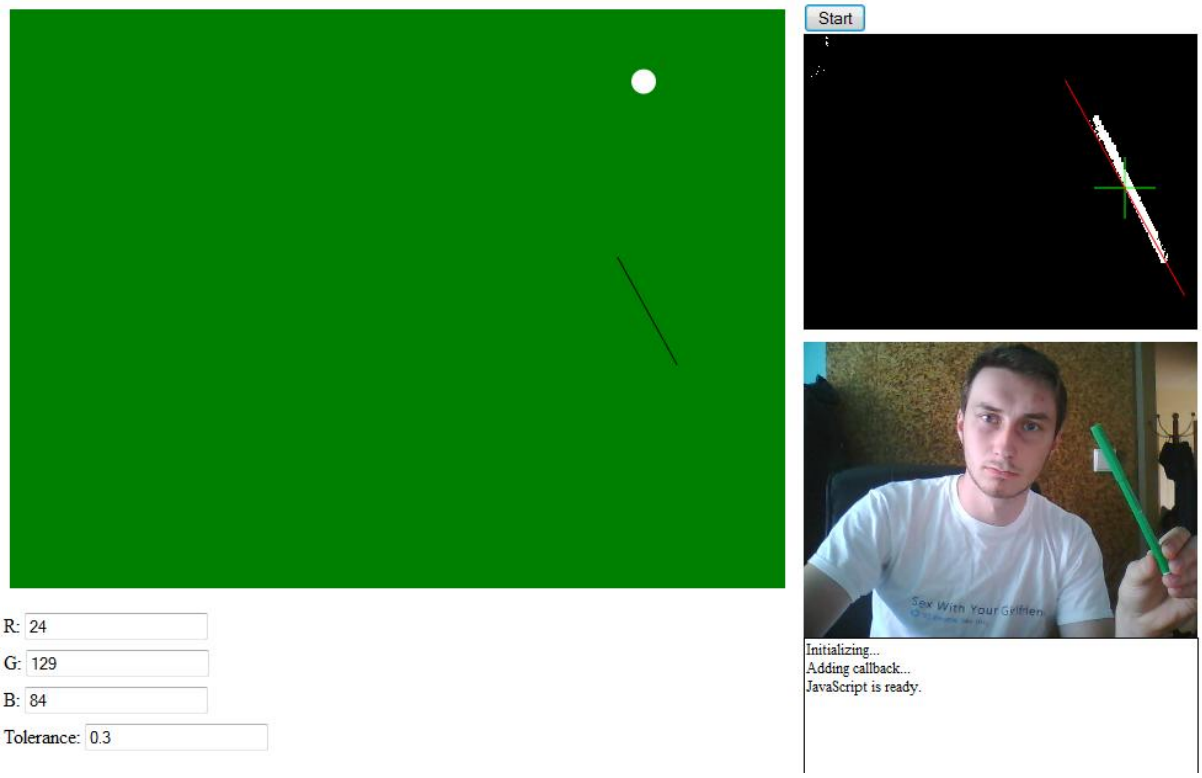
Další věcí, kterou potřebujeme zjistit, je úhel natočení objektu. Vezmeme všechny pixely, které jsou nad úrovní středu (pixely, které mají menší y než vypočtený střed). A spočítáme jejich průměrný rozdíl x od x vypočteného středu. To samé uděláme i pro y. Dostaneme další bod. Úsečka mezi těmito body má stejný úhel jako náš objekt. Úhel dopočteme snadno, pomocí pravoúhlého trojúhelníku funkcí tangens. Zvětšená úsečka pod správným úhlem je na obrázku viz výše červená.

4) Vykreslení hrací plochy

Nyní již máme polohu a uhel natočení objektu a můžeme vykreslovat hrací plochu. Plocha se vykresluje pomocí JavaScriptu do HTML objektu canvas. Poloha míčku je dáma směrem letu, jeho předchozí polohou a rychlostí letu. Míček se odráží od stěn a také od našeho objektu (pálky), odráží podle zákona uhlu dopadu a odrazu. Celé scéna se vykresluje po určitých intervalech (20ms) znovu a mění se polohy objektů (míčku a pálky). Míčku podle dráhy letu a pálky podle pohybu našeho předmětu snímaného kamerou. Hrací plocha je zobrazena níže. (Míček je bílý, pálka je černá úsečka.)



Zobrazení kompletního náhledu: (Program se spouští tlačítkem Start)



Závěr:

Zkompletovaná semestrální práce funguje podle našich požadavků. Jediným nedostatkem je přenos dat, který je pomalý. Řešením by mohlo být zpracovat obrázek ve Flashy a odesílat jenom polohu a úhel detekovaného objektu.
Semestrální práce mi dala mnoho nových znalostí. Byla pro mě přínosem, že jsem se naučil pracovat s dalšími nástroji na tvorbu webových aplikací (Flex/Flash, HTML5 – canvas objekt a poznal jsem další možnosti JavaScriptu).