Slovenská technická univerzita
Fakulta informatiky a informačných technológií
Ilkovičova 3, 842 16 Bratislava 4

Zábavný systém pre spolucestujúcich v automobile

Tímový projekt

Projektová dokumentácia

Vedúci tímového projektu: Ing. Vanda Benešová, PhD.
Členovia tímu:
Bc. Patrik Polatsek
Bc. Martin Petluš
Bc. Peter Hamar
Bc. Róbert Sabol
Bc. Jakub Mercz
Bc. Lukáš Sekerák
Bc. Marianna Mušinská
Bc. Juraj Jarábek

Názov tímu: Tím č. 3 (Carlos)
Web: http://labss2.fiit.stuba.sk/TeamProject/2013/team03is-si/
Kontakt: team03.1314@gmail.com
Akademický rok: 2013/2014

Obsah

1. Úvod

Táto dokumentácia vznikla v rámci predmetu Tímový projekt pri práci na projekte s názvom Zábavný systém pre spolucestujúcich v automobile. Spomínaný projekt je riešený tímom č. 3 - Carlos. V tejto kapitole sa nachádza zadanie projektu Carlos, jeho základný koncept a ciele.

1.1 Zadanie projektu

Zábavné a informačné systémy postupne stále viac prenikajú do nášho bežného života, automobily nevynímajúc. Obohatená realita (tiež: rozšírená realita, angl. AR – augmented reality) je jednou z najmodernejších, v súčasnosti vyvíjaných metód interakcie. Jedná sa o rozšírenie vnemu skutočného reálneho sveta o virtuálne prvky vo forme textu, obrazovej, akustickej informácie a pod.

V rámci tímového projektu bude navrhnutý a implementovaný systém, ktorý pomocou kamery sníma okolie za jazdy automobilom, metódami počítačového videnia analyzuje vizuálnu informáciu spolu s GPS informáciou a na základe tejto analýzy vie poskytnúť napr. informácie zaujímavé pre turistu. Malý LED projektor spolu s transparentnou projekčnou fóliou umožňujú prezentovať vygenerované informácie (či už grafické alebo textové) tak, aby pre spolusediaceho bol vytvorený systém obohatenej reality. Interakcia s týmto systémom bude realizovaná predovšetkým pomocou mobilného telefónu, prípadne ak to hardvér umožňuje, priamou interakciou dotykom na fóliu.

1.2 Motivácia a základný koncept projektu

Zábavné a informačné systémy sa stávajú čoraz častejšie súčasťou nášho života. Jedným z najmodernejších a pre človeka najprirodzenejších spôsobov interakcie je obohatená realita, kde reálny svet je doplnený o obrazové a textové virtuálne prvky.

Cieľom tohto projektu je zmeniť bočné okienko automobilu na transparentnú projekčnú plochu, pomocou ktorej bude reálny svet dopĺňaný o ľubovoľné virtuálne informácie so zábavným i náučným zámerom. Na okienku auta nám tak vznikne rozšírená realita, ktorá nás môže informovať o našom bezprostrednom okolí.

Na prezentáciu vygenerovaných informácii použijeme malý LED projekt spolu s

transparentnou projekčnou fóliou. Aplikáciu bude môcť používateľ ovládať veľmi jednoducho – pomocou hlasu a gest s mobilným telefónom.

Obr. 1. Základný koncept riešenia

Pri vývoji tohto programu využijeme programovacie jazyky C/C++, Javu pre mobilnú platformu Android, knižnicu OpenCV a OpenGL, či zariadenie Kinect.

Výsledkom tohto projektu bude prototyp interaktívneho systému pre obohatenú realitu pre spolucestujúceho v automobile.

Podstatou našej aplikácie bude rozoznávať zaujímavé objekty nasnímané kamerou na aute pomocou gps polohy auta a našej internej databázy objektov záujmu. K takto detegovanému objektu doplní do reálnej scény nielen jeho názov, ale aj iné zaujímavé textové i obrazové informácie. Napr. pri pamiatkach zobrazí otváracie hodiny, výšku vstupného a fotografie. Používateľ si na začiatku zvolí kategórie, na ktoré chce byť upozorňovaný ako napr. pamätihodnosti, reštaurácie a kaviarne. Výhodou našej aplikácie bude aj to, že nebude vyžadovať pripojenie na internet.

Aplikácia ale nebude slúžiť len ako vzdelávací systém, ale aj ako zábavný systém pre deti. Jednou z hier, o ktoré bude naša aplikácia doplnená, bude spočívať v uhádnutí názvu objektu, príp. v zodpovedaní otázky, ktorá sa bude týkať tohto objektu.

Naša aplikácia bude pozostávať z viacerých modulov. Mobilné zariadenie bude zaznamenávať naše povely a gestá, zároveň bude zisťovať našu aktuálnu gps polohu. Následne sa spracuje snímka získaná z kamery. Aplikácia bude rozoznávať objekty s využitím metód počítačového videnia s knižnicou OpenCV. Kinect nám bude slúžiť na detekciu polohy hlavy. Informácie o polohe zdetegovaného objektu a polohy hlavy sa využijú na výpočet, na ktorej pozícii projekčnej fólie sa majú spolusediacemu v aute zobraziť virtuálne prvky. Na samotné vykreslenie informácií, či už textových alebo obrazových využijeme OpenGL, ktoré pomocou projektoru zobrazíme na bočné sklo.

Projekt je podporovaný grantom Nadácie Volkswagen.

Téma zahrňuje i vývoj:

- Výskum a vývoj pokročilých metód počítačového videnia
- Výskum a vývoj pokročilých pokročilých metód počítačovej grafiky
- Výskum a vývoj moderných metód interakcie

1.3 Ciele projektu

Primárnym cieľom projektu je vytvoriť funkčný prototyp interaktívnej aplikácie obohatenej reality, ktorá na bočné sklo automobilu zobrazí doplnkové informácie vzdelávacieho a zábavného charakteru.

Medzi základné ciele u úlohy projektu Carlos patrí:

- presné rozpoznávanie objektov
- vykonávanie aplikácie v reálnom čase
- vypočítanie polohy pre zobrazenie doplňujúcich informácií
- generovanie grafických prvkov rozšírenej reality
- ovládanie systému pomocou Android aplikácie
- ovládanie hry pomocou Android aplikácie
- poskytovanie 2 aplikácií (hier), ktoré využívajú obohatenú realitu
- získavanie GPS a časových značiek v reálnom čase pomocou Android aplikácie
- použitie lokálnej databázy pri rozoznávaní objektov záujmu na snímkach
- vypočítanie polohy hlavy z Kinectu
- integrácia komunikácie a celého systému

2. Popis projektu (1. - 3. šprint)

Popis projektu v tejto kapitole sa vzťahuje na prvé 3 šprinty. Je tu opísaný celkový pohľad na architektúru projektu Carlos. Za ním uvádzame podrobný popis komponentov, z ktorých sa projekt skladá. Ku každej časti sú v závere spomenuté úlohy, ktoré sa k jej vypracovaniu viažu. Prehľad jednotlivých úloh je opísaný po šprintoch v ďalších kapitolách.

2.1 Architektúra systému

2.1.1 Návrh architektúry

Pri projekte Carlos sme sa stretli s rôznymi technológiami a zariadeniami. V rámci požiadaviek sme mali za úlohu použiť Android zariadenie na sledovanie polohy automobilu a ovládanie aplikácie. Ďalšie zariadenie Kinect slúži na sledovanie polohy a rotácie hlavy pre presnejšie mapovanie textu na okne vozidla. Toto okno má za úlohu pomocou projektoru zobraziť rozšírenú realitu. Mali sme použiť aj ďalšie zariadenie - kameru, ktorá mala sledovať okolité prostredie. Z obrazu kamery sme mali za úlohu rozoznať objekty, napríklad ako budovy, hrady a pod.

V projekte teda vystupuje veľa pestrých technológii a ich spojenie má riešiť architektúra. Architektúra je postavená na myšlienke spojiť rôzne zariadenia a rôzne technológie. Ďalšími prioritami pri jej navrhovaní je robustnosť (možnosť rozšírenia komponentov, možnosť nahradiť komponenty inými zariadeniami ako je ďalej vysvetlené), jednoduchosť a čo najväčšia nezávislosť komponentov.

Robustnosť architektúry by nám mala priniesť výhodu najmä v ďalšom semestri, kde by naša architektúra bola pripravená na ďalšie komponenty. Prípadne ak by v tomto semestri nastali komplikácie, architektúra by sa mala jednoducho prispôsobiť.

Jednoduchosť architektúry je dôležitá najmä pre rýchle pochopenie celej podstaty systému, keďže v našom tíme je zopár ľudí, ktorí s týmito technológiami nemali skúsenosti. Preto sa bral ohľad aj na toto riziko.

Nezávislosť architektúry je našou treťou prioritou. Cieľom tejto nezávislosti je vytvoriť jednotlivé komponenty tak, aby boli čo najmenej od seba závislé, čiže aby interface medzi nimi bol čo najjednoduchší. Cieľom bolo zároveň vytvoriť taký celok

komponentu, aby sa bol schopný oň postarať jeden člen tímu, ktorý by mal na starosti návrh, analýzu, implementáciu a testovanie daného komponentu.

Tieto priority sme si určili v prvých šprintoch. O týchto prioritách sme intenzívne diskutovali a až po dohode ich poradí sme ich schválili.

2.1.2 Architektúra a jej komponenty

Jednotlivé komponenty a jej pod časti sme nazvali modulmi, teda vzniklo viacero modulov, kde každý mal definovaný presný vstup, výstup modulu a jeho kompetencie.

Vytvorili sme moduly:

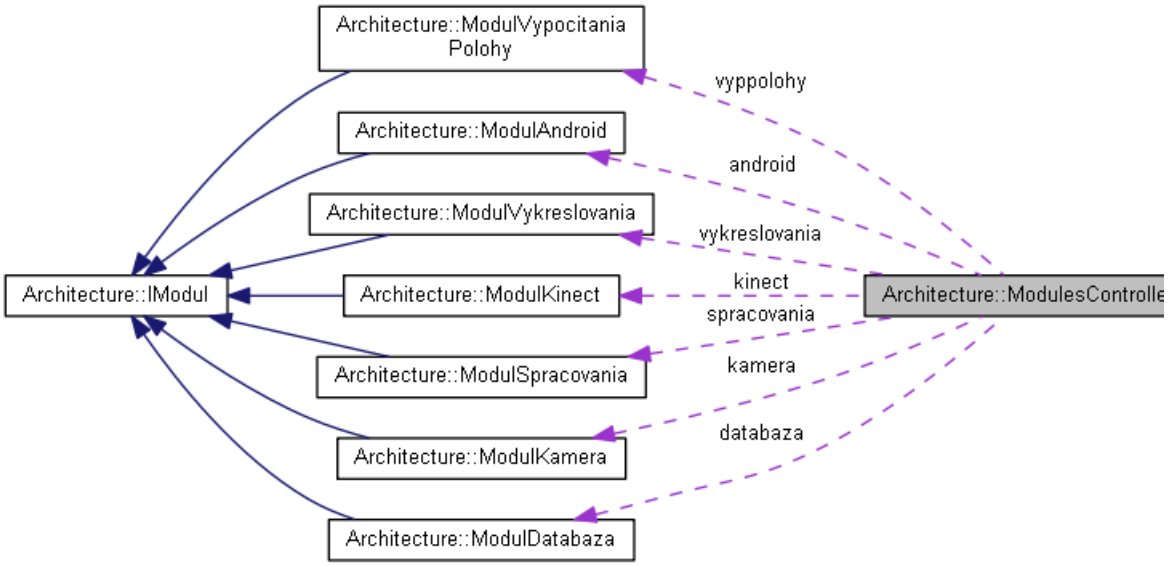
- *Modul spracovania obrazu*
  Vstup: stream z kamery, zoznam možných objektov a cesty k ich fotografiám, maticiam deskriptorov a zoznamom keypointov
  Výstup: poloha zdetegovaných objektov
  Popis: Cieľom je detekcia polohy objektov na snímkach s využitím databázy.
- *Modul vypočítania polohy textu*
  Vstup: GPS, poloha pozerania hlavy v priestore, pozicia detegovaneho objektu, …
  Výstup: poloha textu na okne auta
  Popis: Úlohou tohto modulu je vypočitať polohu textu (prípadne inej obrazovej informácie) na okne auta z definovaných vstupov. Poloha zobrazovanej informácie na okne závisí hlavne od pohľadu používateľa na okno a polohy objektu.
- *Modul databázy*
  Vstup: Zdroj dát v databáze
  Výstup: Namapované entity cez objektový mapovač
  Popis: Cieľom je poskytnúť údaje v databáze cez službu.
- *Modul Android*
  Vstup: Interakcia človeka - gesto, hlas, dotyk na obrazovke; poloha zariadenia
  Výstup: Informácia o vykonanom geste; GPS poloha s časovým údajom
  Popis: Úlohou tohto modulu je slúžiť ako ovládač pre systém. Druhou nie menej dôležitou úlohou je slúžiť ako ovládač pre hry. Tento modul má na starosti taktiež získavanie GPS polohy a času.
- *Modul Kinect*
  Vstup: RGB video stream z Kinectu, stream hĺbkových informácií z Kinectu
  Výstup: poloha bodu pozerania v reálnom priestore

Popis: Modul spracováva údaje z kamier Kinectu a snaží sa nájsť súradnice bodu pozerania pre používateľa. Tieto informácie sú potrebné pre správnu funkcionalitu ďalších modulov a preto ich presnosť ovplyvňuje aj presnosť týchto modulov.

- *Modul rozšírenej reality*
  Vstup: obrázok z videa, obrázok s detegovaným horizontom, vstupy z androidovej aplikácie
  Výstup: zobrazená hra na bočnom skle auta
  Popis: Úlohou tohto modulu je vytvorenie zábavnej hry, ktorú sa bude používateľ hrať pri cestovaní v aute. Hra bude využívať prvky objektov ktoré rozpoznáme.

Modul v konečnom dôsledku mal na starosti určite zariadenie, riadenie určitého zariadenia alebo väčšiu časť logiky aplikácie. Čím sme si tieto moduly mohli ľahko rozdeliť medzi team a pracovať paralelne.

Prvým krokom bolo teda definovanie modulov, tomu sme sa venovali na 1 a 2 stretnutí. Tieto definície sme najprv zapísali do náčrtov, potom diagramov a na konci sme ich implementovali.



Obr.1. Časť class diagramu, ktorý zobrazuje interface IModul a napojenie na ModulesController.

Každý modul dedí od „IModul“, čo je interface. Každý modul môže mať rôzne vstupy a

výstupy. Priamo vstupy a výstupy sú zabalené v triede, ktoré sme nazvali ,,NazovModulu::In“ a ,,NazovModulu::Out“. To sme spravili, preto aby tieto vstupy a výstupy mohli byť robustné, teda ak budeme chcieť neskôr poslať novú informáciu do modulu, stačí pridať do atribútu danej triedy novú informáciu. Zároveň všetky vstupy a výstupy sa vo vlastných triedach serializujú, to chceme využiť neskôr pri debugovaniía možnosti uložiť stav aplikácie. Atribúty serializujeme do textu, ktorý vypisujeme do pomocnej konzole, čím sme si vytvorili možnosť efektívneho sledovania aplikácie a robiť záznamy o jej vykonávaní.

Mnohé moduly si medzi sebou posielajú objekty, obrázky, texty a pod. Pre tieto entity sme si vytvorili vlastne štruktúry, ktoré sú raz definované a využívajú sa vo všetkých moduloch. To pre to, lebo pri analýze technológii sme zistili, že viaceré moduly budú využívať rôzne knižnice a cieľom bolo prenechať knižnicu len danému modelu. Respektíve každému modulu si ponechať svoju knižnicu a prepojiť celý systém cez naše interfaci a naše entity.

2.1.3 Hierarchia architektúry

Pri druhom stretnutí sme sa rozhodli využiť určite metodiky pomenovania súborov a vytvoriť prehľadnú hierarchiu súborov a zložiek. V našom prípade sme použili C++ teda pre každú triedu sme vždy definovali header a cpp súbor. Kde header obsahuje presnú definíciu a cpp subor obsahuje implementáciu.

Moduly sú zadefinované vo vlastnom projekte a obsahujú svoje balíky. Názvy projektov:
com.carlos - riadiaci modul, hlavný proces a projekt
com.carlos.kinect - modul pre určenie polohy hlavy a bodu pozerania
com.carlos.tcp - modul tcp, stará sa o vytvorenie tcp servera
com.carlos.spracovanieReality - modul pre určenie polohy zdetegovaných objektov
com.carlos.vystupObrazovky - modul vykresľovania reality
com.carlos.db - modul pre databázu
com.carlos.CalculateTextPosModule - modul pre výpočet polohy textu na okne

Jednotlivé projekty zase obsahujú balíky, napríklad pre riadiaci modul sú:
architecture - entities
- modules
db - obsahuje zdrojové kódy pre prácu s databázou
main - obsahuje hlavný algoritmus a riadenie aplikácie

2.1.4 Riadiaci modul

Projekt com.carlos,teda riadiaci modul je špecifický tým, ze spája všetky moduly a riadi ich. Stará sa tiež o paralelný beh modulov, pre posielanie správ a údajov. O toto pre posielanie sa stará trieda ModulesController, v súbore class.ModulesController.cpp.

Každy modul robí jeden člen tímu. Po jeho otestovaní a dopísaní dokumentácie ma za úlohu exportnuť tento modul do DLL - dynamic link library, kde tento export si pripojí naša hlavná aplikácia a začne ho používať.

V riadiacom module sa nachádza aj hlavný algoritmus. Ten sa nachádza v triede Carlos, trieda je v súbore class.Carlos.cpp. Hlavný algoritmus má prístupnené moduly cez spomínaný ModulesController. Stream z kamery sa delí na jednotivé snímky, snímok je vstupom do tohto algoritmu. Následne celý algoritmus zbehne a výstupom je rozšírena realita na okne auta. Pre lepšiu ilustráciu vkládame algoritmus:

```
void Carlos::spracujJedenSnimok(Image& image) {
    // Z gps suradnic sa musi synchronizovane pockat, potom sa moze ist dalej
    // Lebo ked snimka meska, tak gps moze byt uz o par metrov dalej
    GPS gps = controller->android->getGPS();
    Rotation rotaciaHlavy = controller->kinect->getAktualnaRotaciaHlavy();

    // Modul preprocessingu
    vector<WorldObject> recepts = controller->databaza->najdiVsetkySvetoveObjektyBlizkoGPS(gps);

    // Modul spracovania
    ModulSpracovania::In spracovanie;
    spracovanie.image = image;
    spracovanie.recepts = recepts;
    ModulSpracovania::Out vysledokSpracovania = controller->spracovania->detekujObjekty(spracovan

    // Modul vypoctu polohy
    vector<Position> najdenePozicie; // synchronizovane
    for(uint i=0; i < vysledokSpracovania.objects.size(); i++) {
        ModulVypocitaniaPolohy::In vypocetPolohy;
        vypocetPolohy.gps = gps;
        vypocetPolohy.polohaObjektu = vysledokSpracovania.objects.at(i).position;
        vypocetPolohy.rotaciaHlavy = rotaciaHlavy;

        ModulVypocitaniaPolohy::Out polohaTextu;
        polohaTextu = controller->vyppolohy->vypocitajPolohuTextu(vypocetPolohy);
        if(polohaTextu.najdeny) {
            najdenePozicie.push_back(polohaTextu.polohaTextu);
        }
    }

    // Modul vykreslovania
    ModulVykreslovania::In vykreslovanie;
    vykreslovanie.image = image;
    vykreslovanie.najdenePozicie = najdenePozicie;
    controller->vykreslovania->vykresliObrazokSRozsirenouRealitou(vykreslovanie);
    imshow("Test", image.data);
}
```

2.1.5 Súvisiace úlohy

#57 [Architektura] Spracovanie sprav z Androidu
#35 [Architektúra] Fake ovládanie
#56 [Architektura] Riadiaci modul
#40 [Architektura] Prvotna analyza architektury
#52 [Architektura] Vygenerovanie diagramov a dokumentacie
#14 [Architektura] Nacitanie videa
#10 [Architektúra] Analýza a definovanie interfaces
#21 [Architektura] Definovanie tried pre moduly
#22 [Architektura] Nacitavanie DLL suborov
#23 [Architektura] Hlavny algoritmus
#24 [Architektura] Prototyp
#23 [Architektura] Thread cast

2.2 Modul spracovania obrazu

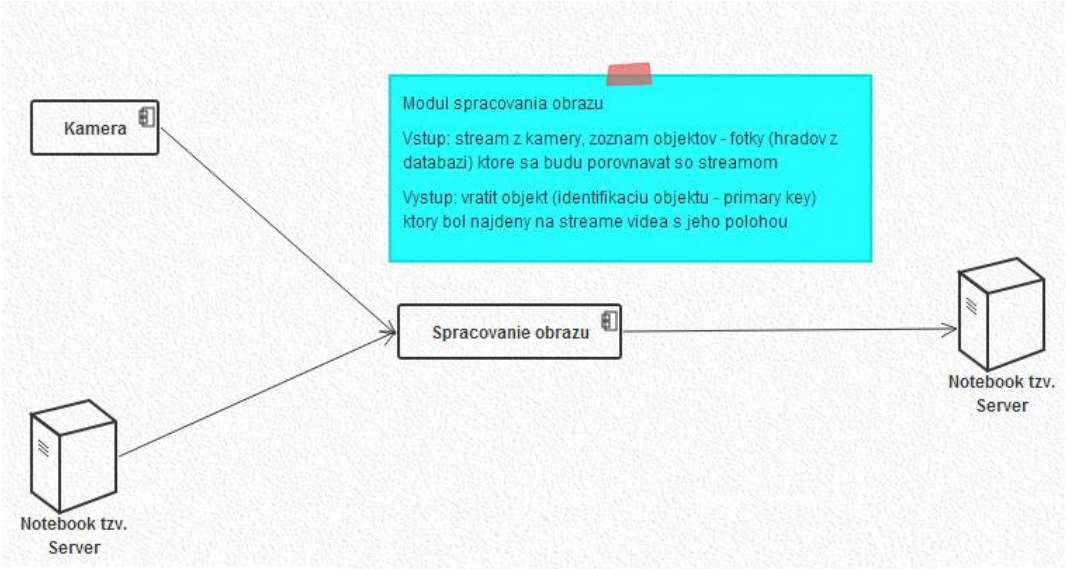
Modul spracovania obrazu slúži vo všeobecnosti na spracovanie snímok prichádzajúcich z webkamery. Jeho výstupom je v súčasnosti poloha detegovaných objektov (POI), ktoré sa porovnali s obrázkami uložených v lokálnej databáze.

2.2.1 Globálny cieľ

Vytvoriť aplikáciu, ktorá bude rozoznávať objekty zo streamu z webkamery a vracať polohy rozoznaných objektov v reálnom čase.

2.2.2 Cieľ na zimný semester

Vytvoriť prototyp aplikácie, ktorý vráti polohu najpravdepodobnejšieho objektu zo streamu z webkamery.



Obr. 1. Modul spracovania obrazu

2.2.3 Vstupy

- stream z kamery
- zoznam možných objektov a cesty k ich fotografiám, maticiam deskriptorov a zoznamom keypointov

2.2.4 Výstupy

- polohy zdetegovaných objektov

2.2.5 Analýza

OpenCV ponúka viaceré možnosti na detekciu objektov. Zanalyzoval som a zvážil som výhody a nevýhody viacerých prístupov:

- *spätná projekcia* → vytvorenie histogramu zo známeho objektu a následné vytvorenie spätnej projekcie na stream z kamery nám vytvorí pravdepodobnostnú mapu, ktorá nám ukáže ako dobre pixely zo streamu zapadajú do rozloženie farieb z nášho histogramu objektu → táto metóda je veľmi citlivá na zmeny osvetlenia a mnoho objektov záujmu (POI) má veľmi podobné histogramy, preto je táto metóda nevhodná
- *template matching* → template, v tomto prípade obrázok nášho POI sa metódou sliding window hľadá na streame z kamery, táto metóda je tak isto nevhodná, veľmi citlivá, pretože objekt môže byť napr. inak natočený a detekcia by už mohla byť neúspešná
- *deskriptory* → metóda spočíva v extrakcii dobre viditeľných bodov (keypoints) na obrázku s POI a streamu z kamery, potom sa okolie týchto bodov opíše pomocou deskriptorov (n-dimenzionálne vektory). Výhodou je že deskrirptory sú vo väčšej miere invariantné voči zmenám jasu a deformácii obrazu oproti predchádzajúcim metódam. Deskriptory objektu a stremu sú porovnávané - pre každý keypoint objektu sa nájde jeho najlepšia zhoda na streame.

Z týchto prístupov sme vybrali metódu detekcie pomocou deskriptorov ako najlepšiu.

2.2.6 Návrh

Pri návrhu modulu spracovania obrazu sa použili niektoré časti kódu od M. Račeva.

Modul bude príjmať stream z videa a množinu možných kandidátov na objekty na základe gps, ktoré na obraze môže modul zdetegovať. Ku každému kandidátovi na objekt dostane zároveň cestu k fotografii/fotografiám objektu, množine keypointov a matici deskriptorov, kde každý riadok opisuje okolie jedného bodu. Deskriptory budú vo finálnej fáze pre všetky objekty predpočítané, aby sa zvýšila rýchlosť algoritmu. Fotografie spolu s keypointami a deskriptormi sa načítajú pre všetky objekty. Pre stream z kamery nájdeme keypointy, ktoré sa opíšu rovnako pomocou deskriptorov.

Deskriptory POI a streamu sa v ďalšej fáze porovnajú a nájdu sa pre každý keypoint objektu k nemu najbližšie 2 keypointy na streame z kamery. Porovnávanie deskriptorov

je založené na ich vzájomnej euklidovskej vzdialenosti (L2).
Výsledné zhody (match) sa v ďalšej fáze prefiltrujú:

- L2 medzi deskriptorom bodu objektu a jeho najbližším bodom na streame (match[i][0]) musí byť výrazne menšia než L2 s jeho 2. najbližším bodom na streame (match[i][1]) →
  `distance(match[i][0]) < threshold * distance(match[i][1])`
- nájdeme ku každému bodu objektu jeho najbližší na streame, ale zároveň to urobíme aj opačne, t.j. pre každý bod na streame nájdeme k nemu najbližší, ak sa tieto zhody rovnajú, tak match akceptujeme →
  `match12[i].query == match21[i].train`

Pre každú fotografiu objektu sa pokúsime nájsť homografiu - maticu perspektívnej transformácie (H) pomocou RANSAC metódy a eliminujeme nevyhovujúce zhody (outliers). Tie zhody, ktoré ostali nazveme dobré zhody (good_matches). Následne určíme pomer: good_matches / matches, priemernú L2 všetkých good_matches a veľkosť zdetegovaného objektu. Z kandidátov na objekty vyberie tie, ktoré splnia hraničné hodnoty týchto podmienok. Tieto objekty považujeme za nájdené na streame.

Obr. 2. Nájdenie ”dobrých” zhôd medzi objektom a scénou.

Polohu zdetegovaného objektu určíme pomocou matice H a nájdenú pozíciu objektu modul predá na výstup ďalej na spracovanie.

Obr. 3. Detegovaný objekt

Zdetegované objekty bude možné trackovať pomocou Lucas-Kanade sledovača, ktorý nájde pozíciu bodov/objektov na nasledovnom obrázku.

2.2.7 Implementácia

Modul bol implementovaný v C++ s knižnicou OpenCV a SiftGPU. Ako deskriptory boli zvolené:

- SURF deskriptory (OpenCV)
- SIFT deskriptory vypočítavané na GPU (SiftGPU)

V súčasnosti ešte nie je implementované sledovanie objektov. Deskriptory objektov ešte nie sú dopredu vypočítané, počítajú sa vždy nanovo. Modul zatiaľ vracia polohu najpravdepodobnejšieho objektu.

2.2.8 Testovanie

Testovanie prebiehalo na vytvorených fotografiách budov ako hrad a kostol za dobrých svetelných podmienok. Pri ďalšom testovaní boli použité fotografie z Google Streetview.

2.2.9 Kalibrácia kamery

Pre správne zobrazovanie objektov je potrebné, aby sa grafické informácie premietané z

projektora zobrazovali čo najpresnejšie. Preto musíme na začiatku skalibrovať kameru pomocou nasledovnej funkcionality.

2.2.9.1 Vstup

- dva obrázky

2.2.9.2 Výstup

- matica perspektívnej transformácie (H)

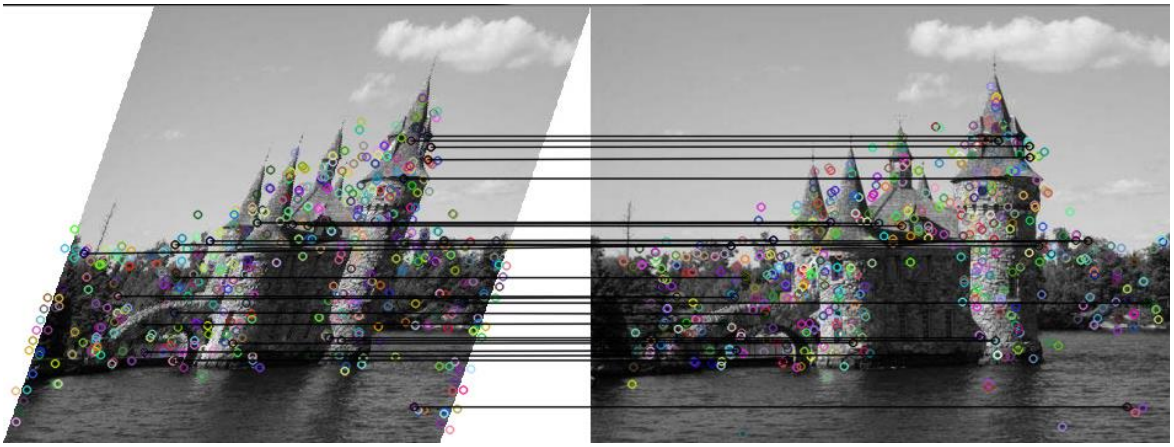
2.2.9.3 Analýza

Technika, ktorá zjednocuje viaceré obrazy, ktoré možu byť navzájom posunuté do jedného sa nazýva registrácia obrazu. Kalibrácia kamery registruje obraz z virtuálneho sveta s reálnou scénou nasnímanou s videokamerou.

Na registráciu obrazu použijeme opäť deskriptory.

2.2.9.4 Návrh

Funkcia kalibrácie kamery nájde maticu perspektívnej transformácie (H) rovnakým postupom ako v module spracovania obrazu. Tá sa potom využije na “prekrytie” oboch obrazov.



Obr. 4. Nájdené zhody medzi 2 obrázkami (1. obrázok je zdeformovaný)

Obr. 5. Transformovaný obrázok na základe matice prespektívnej transformácie H.

2.2.9.5 Implementácia

Táto funkcionalita bola implementovaná v C++ s knižnicou OpenCV. Ako deskriptory boli zvolené SURF deskriptory (OpenCV).

2.2.9.6 Testovanie

Testovanie prebiehalo na 2 rovnakých obrázkoch, pričom jeden bol zdeformovaný.

2.2.10. Súvisiace úlohy

#38 [OpenCV] preštudovať dostupné metódy na detekciu objektov
#1 [OpenCV] prvý prototyp modulu detekcie objektov
#27 [OpenCV] analyzovať a použiť kódy M. Raceva
#4 [OpenCV] detekcia objektov so streetview fotkami
#29 [OpenCV] rozoznanie objektu na obrázku
#17 [OpenCV] registrácia obrazu
#46 [OpenCV] úprava spracovania/registrácie obrazu

2.3 Modul Kinect

Pre dosiahnutie správneho umiestnenia pri zobrazovaní objektov je potreba určiť bod, z ktorého sa používateľ na okno pozerá. Tento modul má za úlohu nájdenie polohy hlavy používateľa a výpočet súradníc spomínaného bodu v priestore.

2.3.1 Globálny cieľ

Vedieť vypočítať bod pozerania používateľa s čo najväčšou presnosťou a aj za nepriaznivých podmienok.

2.3.2 Cieľ na zimný semester

Vedieť odhadnúť bod pozerania používateľa bez potreby videnia tváre.

2.3.3 Vstup

- RGB video stream z Kinectu
- stream hĺbkových informácií z Kinectu

2.3.4 Výstup

- poloha bodu pozerania v reálnom priestore (súradnice x, y, z)

2.3.5 Analýza

Pre určenie polohy hlavy a následne aj hľadaného bodu sme uvážili ako vhodný nástroj Kinect, ktorý poskytuje ako farebný obraz, tak aj hĺbkovú informáciu, vhodnú pre výpočet súradníc bodu a tiež pre niektoré techniky rozpoznávania objektov. Pre zjednodušenie práce s týmto nástrojom sme si vybrali knižnicu freenect, súčasť projektu OpenKinect.org.

Samotné detekovanie polohy hlavy, tváre a potrebného bodu je vykonávané cez knižnicu OpenCV pomocou kaskádových klasifikátorov vhodných na detekciu objektov vo videu.

2.3.6 Návrh

Tento modul bude prímať z kinectu dáta obrazové aj hĺbkové a to pomocou spomenutej knižnice freenect. Tá tieto dáta zachytí a spracuje a následne odošle na spracovanie knižnicou OpenCV, ktorá nájde tvár. Na konci modul odhadne bod pozerania na tvári.

2.3.7 Implementácia

Implementácia prebehla v C++ pomocu knižníc freenect a OpenCV. Ako pomocný súbor pre rozpoznávanie tváre bol použitý súbor haarovych kaskád.
Vstupom pre tento modul je samotný obraz z kinectu a to vrátane hĺbkovej mapy, posunutej aby sedela s obrazom.
Výstupom sú súradnice x, y, z bodu z ktorého sa používateľ pozerá.

Aktuálne modul podporuje len detekciu tváre a určenie bodu pomocou odhadu (polovica šírky tváre, ⅔ výšky). V ďalšom vývoji je plánované nájdenie celej hlavy a teda detekcia bez videnia tváre a tiež spresnenie nájdených súradníc.

2.3.8 Testovanie

Modul bol úspešne testovaný na 4 tvárach, kde približne dokázal určiť potrebný bod. Tiež bola testovaná presnosť hĺbkovej mapy na daný bod pomocou ručných meraní ozajstnej vzdialenosti.

2.3.9 Súvisiace úlohy

#43 [Kinect] nainštalovať knižnicu pre Kinect
#08 [Kinect] hĺbkova informácia
#33 [Kinect] detekcia tváre
#50 [Kinect] Výpočet súradníc bodu z pohľadu kinectu
#44 [Kinect] Rozpoznávanie hlavy

2.4 Modul vypočítania polohy textu

Pri zobrazovaní informácií o detegovanom objekte na premietacie plátno (okno vozidla) je potrebné rátať s viacerými vstupmi a pozíciu informácie na okne, prípadne nejakého virtuálného objektu premietaného na okno je potrebné upravať podľa toho v akom uhle sa práve na daný detegovaný objekt pozerá daný človek cez okno a ako ďaleko je od okna. Úlohou tohto modulu je práve vypočítanie správnej pozície informácie alebo objektu na okne vozidla

2.4.1. Globálny cieľ

Vytvoriť modul, ktorý bude rešpektovať všetky dole vymenované vstupy a bude správne vypočítavať polohu text aj pri prudkých zmenách rýchlosti auta a prudkých zmenách smeru auta.

2.4.2. Cieľ na zimný semester

Vypočítavať polohu textu pre pohybujúce sa auto, pričim nepočítať s prudkými zmenami rýchlosti alebo prudkých zmien smeru auta.

2.4.3. Vstupy

- GPS objektu
- GPS auta
- vzdialenosť kamery od okna
- pozícia kamery v okne v rámci x
- (x1,y1),(x2,y2) = dva rohy (lavý horný a pravý dolný) detegovaný objekt na okne
- (x,y,z) hlavy = bod pozerania hlavy v priestore

2.4.4. Výstupy

- (x,y) - stred polohy pre informáciu o danom objekte

2.4.5. Analýza

Pre vypočítanie správnej polohy informácie na okne sme identifikovali využitie nasledovných vstupov:

1. GPS objektu

2. GPS auta
3. vzdialenosť kamery od okna
4. pozícia kamery v okne v rámci x
5. (x1,y1),(x2,y2) = dva rohy (lavý horný a pravý dolný) detegovaný objekt na okne
6. (x,y,z) hlavy = bod pozerania hlavy v priestore

Modul nebude využívať žiadne externé knižnice a jeho výstupom, bude len jeden bod (x,y) - poloha v rámci okna auta, kde sa nachádza stred polohy informácie pre daný detegovaný objekt. Stred súradnicovej sústavy bude najlepšie umiestniť do lavého horného rohu okna auta (premietacieho plátna). V prvej implemntácii tohoto prototypu sme sa zameráme len na ideálnu situáciu a to takú, že vozidlo sa pohybuje po priamke a neuvažujeme žiadne zákruty ani prudké zmeny rýchlosti, ktoré by mohli v značne krátkom časovom okamihu ovplyvniť polohu informácie.

2.4.6. Návrh

Vstupy 1. a 2. súžia na vypočítanie priamej vzdialenosťi objektu od auta na náslendé ďalšie využie tejto vzdialenosti v algoritme. Vstupy 3. a 4. slúžia na to, aby sme vedeli správne identifikovať uhol, pod akým sa používateľ pozerá na objekt a tak správne vypočítať kolmú vzdialenosť objektu od vozidla spolu s predchádzajúcou vypočítanou priamou vzdialenosťou. Vsup 5. je už samotná pozícia objektu na okne zistená algoritmom na rozpoznávanie objektov. Všetky tieto vypočítané dáta využijeme už pri vypočítaní samotnej polohy informácie spolu so vstupom 6. o polohe pozerania hlavy v priestore.

2.4.7. Implementácia

Popísaný algoritmus je implementovaný v programovacom jazyku C++ bez využitia akýchloľvek externých knižníc. Výstupom algoritmu je bod (x,y), kde na okne sa nachádza stred polohy pre informáciu o danom objekte.

2.4.8. Testovanie

Testovanie prebiehalo vytvorením 3D scény v programovacom jazyku Java Script s využitím knižnice Processing.js. 3D scéna obsahovala improvizované okno vozidla, hlavu človeka a detegovaný objekt. Úlohou v tejto scéne bolo vypočítanie správnej polohy informácie na okne z už spomenutých vstupov. GPS v tomto prípade bolo vynechané z výpočtov a namiesto toho sme využili pozíciu v objektu a vozidla priestore. Testovanie na tejto scéne prebehlo úspešne a algoritmus bol po tomto úspešnom otestovaní

prepísaný do jazyka C++, aby sme ho mohli využiť v našom projekte.

2.4.9. Súvisiace úlohy

#5 [Poloha] Modul na výpočet polohy objektov na okne

2.5 Modul Android

Modul ovládania systému, ktorý zahŕňa ovládanie spúštania aplikácii, ktoré systém obsahuje alebo slúži ako ovládač pre hranie hry na projekčnej ploche je inteligentný telefón s operačným systémom Android. v súčasnosti je jeho výstupom TCP soket s konkrétnou informáciou, ktorú sme na ovládači zvolili. Môže to byť pohyb prstu na obrazovke v 4 smeroch - hore, dole, doľava a doprava, druhou možnosťou je nakláňanie telefónu v 4 smeroch a treťou možnosťou je hlasový povel v anglickom jazyku.

2.5.1 Globálny cieľ

Vytvorenie aplikácie, ktorá bude primárne slúžiť na zapínanie / prepínanie medzi funkciami v systéme. Druhou nie menej dôležitou úlohou aplikácie je slúžiť ako ovládač pre hru v systéme. Modul android ma na starosti taktiež získavanie GPS polohy auta s časovou stopou, ktorú bude odosielať iným modulom v systéme.

2.5.2 Cieľ na zimný semester

Prispôsobenie aplikácie Controller, ktorú sme prevzali z minuloročného tímového projektu tak, aby slúžila naším potrebám. Taktiež je potrebné doplnenie ovládania hry s využitím už implementovaných funkcií, ktoré už aplikácia má. Vytvorenie GPS modulu, ktorý bude získavať aktuálnu polohu auta spolu s časovou stopou, ktorá sa v systéme bude využívať.

2.5.3 Vstupy

- gestá - dotyk na obrazovke
- hlas
- nakláňanie zariadenia - gyroskop, akcelerometer
- poloha zariadenia

2.5.4 Výstupy

- informácia o vykonanom geste
- GPS poloha zariadenia s časovou stopou

2.5.5 Analýza

Ako vhodný nástroj resp. technológiu na ovládanie celého systému sme po vzájomnej dohode zvolili Android zariadenie - inteligentný telefón / tablet. Hlavnou výhodou je dotykový displej, na ktorom môžeme implementovať rozpoznávanie rôznych giest a pohybov. Druhou možnosťou je využitie gyroskopu a akcelerometra pre ovladanie pohybov v hre. Inteligentný telefón nám samozrejme ponúka aj iné možnosti, ktoré môžeme využiť ako je napríklad rozpoznávanie hlasu a hlasové povely. Tie sú však v offline režime obmedzené na anglický jazyk, ale pre jednoduché povely - “start”, “run”, “one” … by sme to dokázali využiť, keďže najväčším problémom môže byť výslovnosť a akcent daného používateľa. Ďalšou výhodou tejto platformy je aj to, že časť našeho systému využíva aktuálnu GPS polohu, čo je pomerne jednoducho implementovateľné a môžeme to kľudne zahrnúť do tejto aplikácie.

2.5.6 Návrh

Tento modul bude z telefónu odosielať TCP sokety s informáciou o danom geste, či už je to pohybové, hlasové alebo iné gesto. Aplikácia jednak poslúži ako ovládač pre súštanie rôznych možností v systéme, ale aj ako ovládač hier.

2.5.7 Implementácia

Implementácia prebehla v jazyku Java v prostredí Android Development Tool (Eclipse prispôsobený vývoju pre Android). Ako už bolo spomenuté vyššie čiastočne som využil funkčnú implemntáciu z iného tímového projektu, ktorú som však do značnej miery musel upravovať, aby to vyhovovalo naším potrebám. Implementoval som taktiež TCP komunikáciu, kde Android aplikácia sa správala ako klient a Java Server aplikáciu, ktorá príjmala sokety sa tvárila ako server. Výstupom tejto implementácie je odoslaný soket po vykonaní daného gesta.

2.5.8 Testovanie

Testovanie prebehlo na Android zariadeni Samsung Galaxy SII, ktoré komunikovalo s lokálnym serverom a odosielalo na neho sokety. Sokety boli úspešne prijaté.

2.5.9 Android Aplikácia pre zaznamenávanie GPS polohy

Samostatnú časť modulu tvorí aj aplikácia pre záznam GPS polohy .

V našom projekte je veľmi potrebná informácia aktuálnej GPS polohy našeho auta z dôvodu rozpoznania objektov v danej oblasti. Pre dosiahnutie správnej GPS informácie bolo potrebné vytvoriť nasledovný modul ktorý má na starosti zachytenie správnej GPS informácie. Jedná sa o konkrétne samostatnú android aplikáciu, ktorá má za úlohu získanie súradníc ktoré sa získavajú pravidelne každú sekundu. Táto aplikácia v budúcnosti bude súčasťou celého Android modulu a nebude vysupovať ako samostatná aplikácia.

2.5.9.1 Analýza

Pre určenie GPS polohy našeho auta a jej následné uloženie sme uvážili ako najvhodnejší nástroj/zariadenie použiť Android smartfón. Vätšina dnešných Android telefónov obsahuje GPS prijímač a internetové pripojenie ktoré môže zmenšiť odchýlku polohy nameranú smartfónom od reálnej polohy. Ďalšia výhoda využitia smartfónu spočíva v tom, že sme vedeli dopredu, že smartfón bude použitý na ovládanie celej aplikácie a užívateľ jednoducho musí mať jeden pri sebe. Výhoda zistenia polohy android smartfónom bola aj v samotnej implementácie, pretože Android development obsahuje prehľadné API pre získanie GPS polohy.

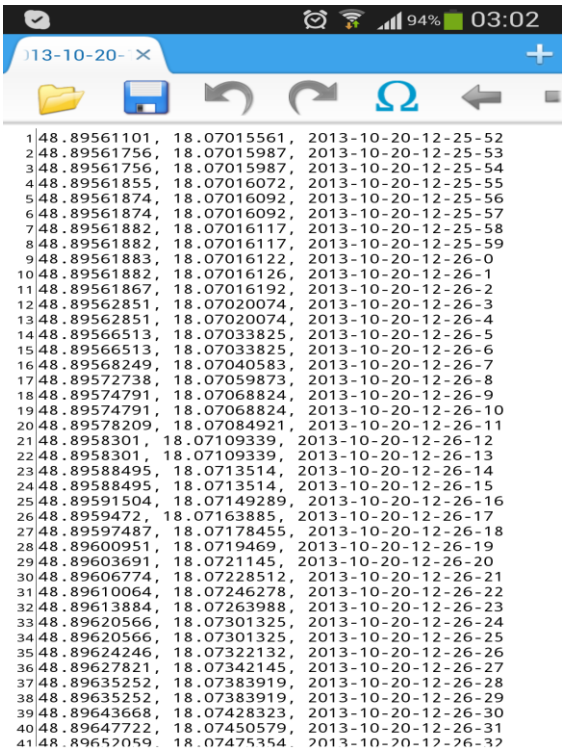
2.5.9.2 Návrh

Aplikácia bude prijímať z GPS prijímača dáta geografického súradnicového systému t.j. informáciu o zemepisnej šírke a zemepisnej dĺžke (z angl. „langitude“ a „longittude“). Keďže naše auto sa občas bude hýbať vačšou rýchlosťou, to znamená naša poloha sa bude zmenou rýchlosti neustále meniť, je potrebné mať stále informácie o aktuálnej polohe, takže sme sa rozhodli získať novú polohu každú sekundu.

2.5.9.3 Implementácia

Implementácia prebehla v Jave (Android SDK) pomocou viacerých knižníc, využili sme najma framework API android.location. Taktiež sme použili triedu LocationManager a Location čo nám podstatne uľahčilo prácu.

Výstupom danej aplikácie je súbor s GPS súradnicami v danom formáte:
(1.stlpec Latitude 2.stlpec Longitute 3.stlpec timestamp)



Obr. 1 Ukážka výstupu aplikácie

Ako je možné vidieť, každý riadok obsahuje informáciu GPS v polohy v danej sekunde. Aktuálna verzia tejto aplikácie zatiaľ podporuje iba lokálne ukladanie GPS informácie v telefóne.

2.5.9.4 Testovanie

Aplikácia bola úspešne testovaná na 4 Android smartfónoch, kde sme dokázali vytvoriť pár testovacích záznamov. Presnosť bola otestovaná s aplikáciou Google Earth. Odchýlka bola zväčša pár metrov.

Obr. 2 Ukážka dizajnu aplikácie

2.5.9.5 Súvisiace úlohy

#36 [Mobil] ukladať gps polohu

2.5.10 Súvisiace úlohy

#41 [Mobil] ukladanie gest
#16 [Mobil] demo pre rozpoznávanie gest
#37 [Mobil] analýza kódov na rozoznávanie gest
#32 [Mobil] TCP komunikácia z Androidu

2.6 Modul rozšírenej reality

Tento modul sa bude starať o zobrazenie hry na bočné okno auta a v neskoršej fáze aj na zobrazenie identifikovaných informácií o viditeľných objektoch. Vykonávať sa to bude za pomoci LED projektora ktorý bude umiestnený priamo v aute.

2.6.1. Globálny cieľ

Vytvorenie minimálne dvoch verzií hier pre používateľov. Prvá bude orientovaná skôr na zábavu. Druhá hra bude mať za úlohu naučiť niečo zaujímavé používateľa a následne ho aj vyskúšať. Taktiež bude potrebné klásť dôraz na to aby sa v hre využívali informácie o rozpoznaných objektoch.

2.6.2. Cieľ na zimný semester

Vytvorenie prvej funkčnej hry, zameranej na zábavu. Pokúsiť sa hru na-implementovať čo najvhodnejšie a tak aby ďalšie rozširovanie - pridávanie hier nebolo problematické. Taktiež dbať aj na používateľa a spracovať hru čo najzábavnejším spôsobom (zaujímavý model, možnosť výberu modelu, rebríčky najlepších...).

2.6.3. Vstupy

- obrázok z videa
- obrázok s detegovaným horizontom
- vstupy z androidovej aplikácie

2.6.4. Výstupy

- zozbrazená hra na bočnom skle auta

2.6.5. Analýza

Keďže cieľom je vytvorenie hry, hľadali sme vhodnú knižnicu za pomoci ktorej by sme túto hru implementovali. Avšak s prihliadnutím na to, že podstatná časť projektu bude implementovaná v jazyku C++ a v IDE Visual studio, rozhodli sme sa zvoliť OpenGL.

Táto knižnica je dostatočne komplexná a existuje pre ňu značné množstvo tutoriálov a dokumentácie. Taktiež použijeme aj ďalšie knižnice ako sú glut, freeglut, glew a mnohé ďalšie ktoré nám uľahčia prácu s OpenGL.

Ďalšia analýza spočívala vo voľbe čo najvhodnejšej hry na implementovanie. Vytvorili sme jednoduchú hru, ktorá formou otázok zisťovala vedomosti hráča a na konci mu zobrazila skóre. Vďaka čomu sme sa lepšie naučili používať knižnicu OpenGL. Taktiež všetci členovia tímu sa mali zamyslieť nad nápadom hry. Takto sme ziskali mnohé nápady. Viac o tom je napísané v samotnom návrhu.

2.6.6 Návrh

Pre zimný semester sme sa dohodli, že vytvoríme hru ktorou by sa mal používateľ baviť. Na stretnutiach sme zvažovali rôzne možnosti hry. Od logických, vedomostných až po letecké. Ako najvhodnejšiu hru sme vybrali lietadlo, ktoré letí v prostredí kde sa nachádza auto. Taktiež sme zvažovali viaceré možnosti ktoré je možné s týmto lietadlom vykonať.

Ako najlepší nápad hry sme vybrali lietadlo letiace nad horizontom, kde je cieľom nahrať čo najviac bodov a dôjsť čo najďalej. Čím používateľ letí bližšie horizontu tím viac bodov dostáva, avšak je aj bližšie pádu. Ak používateľ klesne pod horizont resp. narazí tak hra končí. Počas letu sa hráčovi pripočítavajú body. Na konci sa zobrazí tabuľka, kde si zadá meno a uvidí ako dobre / zle v celkovom rebríčku dopadol. Celá hra by sa mala ovládať pomocou mobilu.

2.6.7. Implementácia

Ako IDE sme si zvolili Visual Studio 2012. Na implementáciu hry použijeme OpenGL a viaceré ďalšie knižnice ktoré nám uľahčujú prácu so samotným OpenGL.

Aktuálne použité knižnice:

- FreeImage
- FreeType
- Glut
- FreeGlut
- Glew
- Glm

Všetky knižnice bolo potrebné u seba si skompilovať a následne ich pridať do modulu rozšírenej reality. V tomto momente už máme načítaný 3D model. Model je otexturovaný. Formát 3D modelu sme použili .obj súbor spojený s .mtl súborom v ktorom sú zaznačené údaje o textúre ktorá sa má použiť na model. Aktuálne sa

dokončilo zobrazenie meniacich sa obrázkov ktoré získame z videa. Scéna už je kompletne vytvorená. Taktiež už je možné hýbať modelom po osiach x a y za pomoci kláves w,a,s,d.

V implementácii sa bude ďalej prokračovať, najviac času zaberie implementácia logiky hry a jej otestovanie, kompletná integrácia s android zariadením, integrácia s modulom spracovania obrazu, ďalej napojenie na databázu, počítanie skóre, prípadne zvažujeme aj možnosť ďať používateľovi na výber z viacerých typov lietadiel.

2.6.8. Testovanie

Bol vytvorený prototyp ktorý spĺňal všetky vytýčené cieľe. Tento prototyp bol otestovaný na všetky možné stavy ktoré môžu v hre nastať. Avšak je dôležité v testovaní pokračovať keďže ešte nie vytvorené všetko.

2.6.9. Súvisiace úlohy

#42 [OpenGL] Naštudovať základy OpenGL
#6 [OpenGL] Odskúšať objavujúce sa a miznúce objekty
#45 [OpenGL] Vypisovanie textu v okne
#18 [OpenGL] Práca na jednoduchej hre
#28 [OpenGL] Pridanie 3D modelu
#34 [OpenGL] Pohyby 3D modelom
#54 [OpenGL] Pripravenie scény za 3D modelom
#55 [OpenGL] Držať lietadlo nad horizontom

2.7 Modul databázy

Modul databázy, zahŕňa inštaláciu databázy, jej kompletnú prípravu. Ďalej vytvorenie dátového modelu, naplnenie ho dátami. V ďalšej fáze ma tento modul sprístupniť databázu pre iné modely v podobe služby. Zároveň tento modul ma vytvoriť možnosť perzistentne uchovávať informácie o rôznych entitách. Ktoré entity sa budú uchovávať sú popísané v dátovom modely. Vytvorenie dátového modelu je sekcii analýza a návrh.

2.7.1. Globálny cieľ

Vytvorenie modulu, ktorý bude mať na starosti riadenie databázy, uchovávanie informácií a jej opätovné získavanie. Celá funkcionalita bude v podobe služby. Služba bude sprístupnená len pre riadiaci modul, ktorý bude preposielať informácie ďalej.

2.7.2 Cieľ na zimný semester

Vytvorenie databázovej služby, ktorá bude spĺňať základnú funkcionalitu. Teda pripojenie na databázu, uloženie informácii a získavanie informácii. Zároveň cieľom je aj vytvorenie prvej verzie databázového modelu a naplnenie fake údajmi pre testovanie. V rámci tohto modulu je potrebné vytvoriť objektový mapovač a poskytnúť údaje v databáze v podobe entít.

2.7.3. Analýza

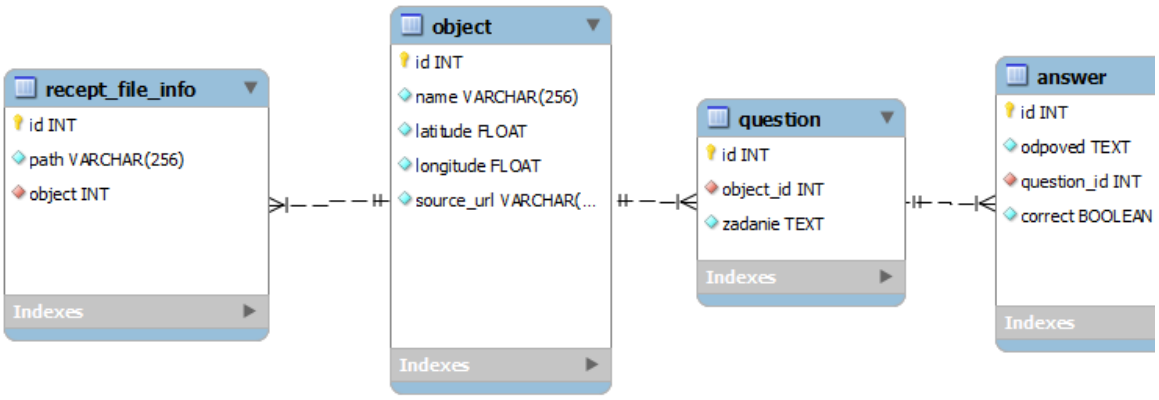
Základom modulu je databáza, takže v rámci analýzy bolo dôležité nájsť databázu, ktorá bude spĺňať naše požiadavky. V rámci analýzy sme sa pozreli na MySql, Oracle, sqlite, postgresql. Všetky databázy okrem sqlite, majú zložitú inštaláciu respektíve majú bežať na servery. Komunikácia nášho programu so serverom by bola komplikovaná a ak by aplikácia bola umiestnená mimo serveru nastali by rôzne komplikácie so sieťovou konfiguráciou a hlavne veľkou latenciou. Predpokladá sa, že naša databáza bude mať 5-10 tabuliek. Pre objektový mapovač sme našli viacero knižníc.

V ďalšej fáze analýzy sme skúmali, aký databázový model potrebujeme. V rámci diskusie medzi členmi sme sa dohodli na modeli a ten uvádzame v kapitole Návrhu.

2.7.4. Návrh

V analýze sme sa v skratke pozreli na dostupné riešenia. Pozreli sme sa na klady a zápory. Nakoniec sme na základe dostupných faktov vybrali sqlite. Najmä preto, že práca s ňou je jednoduchá a pre tento počet tabuliek je postačujúca. Pre tuto databázu existuje C++ API, takže aplikácia sa dokáže priamo napojiť na zdroj dát a priamo pracovať s databázou, bez sieťovej konektivity. Preto databáza môže byť dodaná priamo s aplikáciou.

Mnohé objektové mapovače, ktoré sme našli v procese analýzy boli príliš komplexné pre väčšie systémy. Nakoniec sme vybrali knižnicu sweetql.



Obr.1. Dátový model databázy

2.7.5. Implementácia

Implementácia prebehla v jazyku C++, vo visual studiu prostredí. Pre prácu s databázou bolo použité sqlilite API. Implementácia bola v podstate jednoduchá, spočívala len v prepojení sqlite API s knižnicou sweetql. Zároveň sme tieto dve knižnice zabalili do jednej triedy, do databázovej služby.

V rámci implementácie sme naplnili databázu falošnými údajmi.

2.7.6. Testovanie

Testovanie prebehlo v spolupráci s hlavným modulom, ktorému bola poskytnutá databázová služba. Služba sa pripojila na databázu, úspešne vyhľadala a stiahla údaje z databázy.

2.7.7. Súvisiace úlohy

#53 [Databaza] OOP Mapovac a DB Service
#26 [Databaza] Zadefinovanie entit ktore su v databaze
#25 [Databaza] C++ Wrapper, pripojit sa na databazu
#11 [Databaza] Nainštalovať databázu

3. Prvý šprint

Začiatok šprintu: 23.9.2013

#1 [OpenCV] prvý prototyp modulu detekcie objektov
Zodpovedná osoba: Patrik Polatsek

Na základe analýzy možných spôsobov detekcie objektov sa vytvorí prvý prototyp modulu detekcie objektov. Tento modul sa pokúsi nájsť na obrázku možné objekty pomocou deskriptorov. Na porovnanie sa využije lokálna zložka s.

#2 Vytvoriť webstránku projektu
Zodpovedná osoba: Martin Petluš

Cieľom tejto úlohy bolo vytvoriť web stránku, tímu, ktorá bude slúžiť na prezentáciu projektu (materiálov a informácií k nemu) a zároveň aj jeho členov.

#58 Redmine
Zodpovedná osoba: Martin Petluš

Rozbehať na serveri Redmine, vytvoriť na Redmine projekt k tímovému projektu.

#48 Rozbehanie servera
Zodpovedná osoba: Martin Petluš

Nainštalovať na virtuálny server operačný systém a pripraviť ho na inštaláciu ďalších potrebných programov, ktoré sa budú využívať v tímovom projekte.

#36 [Mobil] ukladať gps polohu
Zodpovedná osoba: Juraj Jarábek

Vytvorenie samostanej GPS aplikácie pre zaznamenávanie GPS polohy spolu s časovým razítkom. Tento submodul resp. aplikácia bude neskôr súčasťou celého Android modulu.

#37 [Mobil] analýza kódov na rozoznávanie gest
Zodpovedná osoba: Róbert Sabol

Stretnutie s kolegom Bc. Richardom Sámelom a prebratie minuloročného projektu Controller. Následne zanalyzovanie aplikácie a pochopenie aké gestá sa dajú využívať.

#38 [OpenCV] preštudovať dostupné metódy na detekciu objektov
Zodpovedná osoba: Patrik Polatsek

Naša aplikácia bude rozoznávať zo snímiek objekty záujmu. Na to je potrebné analyzovať dostupné metódy na detekciu objektov a učiť ich výhody a nevýhody. Z týchto metód je treba vybrať najvhodnejšiu metódu rozoznania objektov pre náš systém.

#39 [OpenCV] nastudovať základy OpenCV
Zodpovedná osoba: Martin Petluš

Naštudovať si základy OpenCV, pre budúcu prácu na projekte. Vyskúšať si jednoduché príklady s použitím knižnice OpenCV. Nainštalovať si OpenCV.

#40 [Architektura] Prvotná analýza architektúry
Zodpovedná osoba: Lukáš Sekerák

V rámci tejto analýzy sme sa pozerali na možnosti architektúry. Akú architektúru vytvoríme a ako budú jednotlivé komponenty podelené a ako budú spolupracovať.

#41 [Mobil] ukladanie gest
Zodpovedná osoba: Róbert Sabol

Prerobenie aplikácie Controller tak, aby dokázala gestá ukladať lokálne na mobilné zariadenie namiesto ukladania na server ako to bolo predpripravené.

#42 [OpenGL] naštudovať základy OpenGL
Zodpovedná osoba: Peter Hamar

V tejto časti analýzi pôjde o dôkladnejšie zoznámenie sa s knižnicou OpenGL. Naštudovanie základnych funkcií, nainštalovanie vývojového prostredia a vytvorenie prvého projektu vo VS s využitím knižnice OpenGL. Taktiež oboznámenie sa s prácou s textúrami, ich načítanie a zobrazenie.

#43 [Kinect] nainštalovať knižnicu pre Kinect
Zodpovedná osoba: Jakub Mercz

Pre uľahčenie prístupu k dátam, ktoré kinect ponúka bude použitá knižnica freenect, ktorú je potrebné správne nainštalovať a dôkladne preskúmať jej používanie.

#45 [OpenGL] Vypisovanie textu v okne
Zodpovedná osoba: Peter Hamar

Úlohou je pridanie textu na obrázok / textúru. Taktiež si otestovať rôzne druhy typov písma a samozrejme poziciovanie textu v okne.

4. Druhý šprint

Začiatok šprintu: 14.10.2013

#3 natočiť videá s gps informáciou
Zodpovedná osoba: Juraj Jarábek

Natočiť niekoľko video záznamov mimo mesta a v meste, ktoré budeme môcť neskôr použiť spolu s gpd informáciou, na ktorú nám poslúži už vytvorená android aplikácia.

#4 [OpenCV] detekcia objektov so streetview fotkami
Zodpovedná osoba: Patrik Polatsek

Doterajšie riešenie, ktoré rozoznáva objekty na obrázkoch treba otestovať na obrázkoch získané z Google Streetview. Toto testovanie je potrebné a záver vyhodnotiť.

#5 [Poloha] modul na výpočet polohy objektov na okne
Zodpovedná osoba: Martin Petluš

Cielom úlohy bolo vytvoriť prototyp, ktorý bude vypočítavať polohu textu na okne. Modul bude počítať len s ideálnou situáciou (nie prudké zmeny rýchlosti ani smeru).

#6 [OpenGL] Odskúšať objavujúce sa a miznúce objekty
Zodpovedná osoba: Peter Hamar

Cieľom tejto úlohy je vytvoriť pohybujúci sa text. Otestovať rôzne druhy zložitejších pohybov a taktiež vyskúšať si zobrazujúci a miznúci text pomocou interpolácie.

#7 [Mobil] demo pre rozpoznávanie giest
Zodpovedná osoba: Juraj Jarábek

Oboznámiť sa s projektom od kolegu Bc. Richarda Sámela. Pochopiť štruktúre projektu a pomôcť Robovi s vytvorením dema pre rozpoznávanie giest.

#8 [Kinect] hĺbkova informácia
Zodpovedná osoba: Jakub Mercz

Táto úloha zahŕňa spracovanie hĺbkovej informácie z kinectu na merateľnú jednotku (milimetre) a jej namapovanie na RGB stream, aby oba vstupy spolu korelovali.

#9 [Architektúra] Definovať aké vstupy a výstupy má každý modul
Zodpovedná osoba: všetci členovia tímu, ktorí pracujú na moduloch

Väčšina členov tímu pracuje samostatne na svojich moduloch. Pre potrebu vytvorenia komplexnej architektúry projektu je ale potrebné zadefinovať vstupy a výstupy každého modulu. Tie sa následne využijú na vytvorenie prvého návrhu architektúry.

#10 [Architektúra] Analýza a definovanie interfaces
Zodpovedná osoba: Lukáš Sekerák

Po tom čo sme definovali vstupy a vystupy modelov. Je potrebne zadefinovať interfaci medzi modulmi. Je potrebné zadefinovať aj základne entity ktoré vystupujú pri komunikácií modulov. Zároveň to všetko treba implementovať.

#11 [Databáza] Nainštalovať databázu
Zodpovedná osoba: Lukáš Sekerák

V rámci modulu databázi, je potrebné spraviť prvý krok a to vytvoriť sqlite databázu.

#16 [Mobil] demo pre rozpoznávanie gest
Zodpovedná osoba: Róbert Sabol

Ukázať aplikáciu a oboznámiť zvyšok tímu s tým ake gestá je možné využiť a aké možností pre ovládanie systému modul Android poskytne.

#17 [OpenCV] registrácia obrazu
Zodpovedná osoba: Patrik Polatsek

Pre správnu funkčnosť celého systému je dôležité, aby boli dolnkové informácie zobrazované na bočnom skle čo najpresnejšie. Preto je na začiatku potrebné skalibrovať kameru. Prístup, ktorý sa pri tom využije sa nazýva registrácia obrazu, ktorá sa pokúsi “prekryť” viaceré obrázky do jedného.

#18 [OpenGL] Práca na jednoduchej hre
Zodpovedná osoba: Peter Hamar

Keďže v projekte plánujeme vytvoriť hru, je potrebné vytvoriť nejaký jednoduchší prototyp hry na to aby sme sa naučili lepšie pracovať s OpenGL. Navrhli sme hru s náučným kontextom. Zobrazovalo sa 10 obrázkou, zobrazila sa otázka ktorá sa týkala objektu na obrázku spoločne s 3 možnosťami a úlohou bolo vybrať správu. Hráčovi sa postupne počítalo skóre.

#20 [TP Cup] Prihláška do súťaže
Zodpovedná osoba: Patrik Polatsek

Náš tím sa rozhodol zúčastniť v súťaži TP Cup 2014. K tomu je potrebné vypracovať prihlášku, kde sa uvedie základný koncept a motivácia k projektu.

#27 [OpenCV] analyzovať a použiť kódy M. Račeva
Zodpovedná osoba: Patrik Polatsek

Na detekciu objektov na obrázku je možné použiť diplomovoú prácu M. Račeva a jeho kódy. Tie treba analyzovať a podľa možností zapracovať do modulu spracovania obrazu.

#28 [OpenGL] Pridanie 3D modelu
Zodpovedná osoba: Peter Hamar

Keďže v hre plánujeme použiť 3D objekty bolo dôležité nejaký objekt načítať. Po preštudovaní formátov 3D modelov sme sa rozhodli použiť .obj súbor. Bolo potrebné skompilovať viaceré knižnice ktoré sme následne použili. Výsledkom bol pridaný 3D model lietadla s namapovanou textúrou.

#29 [OpenCV] rozoznanie objektu na obrázku
Zodpovedná osoba: Patrik Polatsek

Modul spracovania obrazu sa pokúsi rozoznať možné objekty na snímke. V tejto fáze je potrebné vyselektovať z možných kandidátov na objekty tie, ktoré sú skutočne na obrázku.

#31 [TCP] TCP modul do C++ projektu
Zodpovedná osoba: Lukáš Sekerák

V neskoršej fáze vývoja, sme sa zistili, že nie je dobré ak android komunikuje priamo s riadiacim modulom. Preto sme sam rozhodli ďalší modul, ktorý bude mať funkcionalitu TCP servera. Na tento server sa pripojí android a bude mu posielať správy. TCP server v prípade prijatia správy

prepošle správu riadiacemu modulu. V tejto úlohe ide hlavne o riešenie týchto problémov a ich implementáciu.

#32 [Mobil] TCP komunikácia z Androidu
Zodpovedná osoba: Róbert Sabol

Keďže modul Android bude so systémom komunikovať cez TCP server, je potrebné implementovať komunikáciu Android modulu s TCP serverom, ktorému modul Android odošle informáciu o vykonanom geste v tvare TCP soketu.

#33 [Kinect] detekcia tváre
Zodpovedná osoba: Jakub Mercz

Pre prvú verziu detekcie bodu pozerania je potreba detekovať tvár a v tej určiť odhadom bod pozerania.

5. Tretí šprint

Začiatok šprintu: 4.11.2013

#30 Doxygen
Zodpovedná osoba: Juraj Jarábek

Naštudovať Doxygen tutoriály a vygerovať vzorovú HTML dokumentáciu. Taktiež vytvoriť súbor pravidiel dokumentovania kódu pre všetkých členov tímu.

#34 [OpenGL] Pohyby 3D modelom
Zodpovedná osoba: Peter Hamar

Cieľom je vytvoriť pohyby 3D modelu, aby lietadlo simulovalo let. Zatiaľ pomocou klávesnice. Snažiť sa o dosiahnutie čo najrealnejšieho pohybu lietadla.

#35 [Architektúra] Fake ovládanie
Zodpovedná osoba: Lukáš Sekerák

Keďže android modul, nie je ešte priprvený rozhodli sme sa nahradiť android “dump” modulom. Ktorým by sa tiež ovladalá aplikácia ale len za pomoci klávesnice.

#46 [OpenCV] úprava spracovania/registrácie obrazu
Zodpovedná osoba: Patrik Polatsek

Modul spracovania obrazu a funkciu registráciu obrazu je potrebné nateraz sfinalizovať a zapojiť do celkového projektu.

#51 [OpenCV] detekcia horizontu
Zodpovedná osoba: Patrik Polatsek

Hra s lietadlom bude spočívať v tom, že lietadlo nesmie padnúť pod horizont. Na to je potrebné horizont na obrázku zdetegovať. Táto úloha vyžaduje analýzu možných detekcií horizontu a výber najvhodnejšej metódy pre tento projekt.

#54 [OpenGL] Pripravenie scény za lietadlom
Zodpovedná osoba: Peter Hamar

Na to aby sme mohli verne simulovať let lietadla je potrebné pripraviť scénu. Cieľom je zobrazenie videa priamo za modelom lietadla. Video sa bude zobrazovať za pomoci obrázkov resp. textúr v reálnom čase. Taktiež už bude možný pohyb po scéne.

#55 [OpenGL] Držať lietadlo na horizontom
Zodpovedná osoba: Peter Hamar

Cieľom je držať lietadlo nad horizontom. Lietadlo prirodzene klesá k zemi a hráč ho stlačením klávesy dvíha vyššie. Horizont v tomto bode ešte nebude reálny, len nejaký ukážkový.

#50 [Kinect] Výpočet súradníc bodu z pohľadu kinectu
Zodpovedná osoba: Jakub Mercz

Cieľom je preštudovať a implementovať výpočet súradníc v priestore na základe súradníc na obrazovke a hĺbkovej informácie.

#44 [Kinect] Rozpoznávanie hlavy
Zodpovedná osoba: Jakub Mercz

Pre detekciu aj v prípade že nie je možné detekovať tvár, je treba implementovať detekovanie celej hlavy z hĺbkovej mapy.

#44 Inštalácia gerritu
Zodpovedná osoba: Martin Petluš

Nainštalovať na server program gerrit pre tímové code review.

#52 [Architektúra] Vygenerovanie diagramov a dokumentácie
Zodpovedná osoba: Lukáš Sekerák

V rámci tohto šprintu vznikla potreba vygenerovanie diagramov a dokumentácie zo zdrojovéko kódu. V tejto úlohe zodpovedný člen si mal naštudovať možnosti ako generovať dokumentáciu z CPP kódu. Výsledné diagramy mali byť použité na ďalšom stretnutí pri prezentácii.

#53 [Databáza] OOP Mapovač a DB Service
Zodpovedná osoba: Lukáš Sekerák

V tejto úlohe bolo potrebné dokončiť OOP Mapovač, teda dodefinovať všetky entity, dokončiť a otestovať implementáciu OOP Mapovača. Ďalej bolo potrebné celú logiku databázového modulu zakryť do služby, ktorá by bola prístupná ostatným modulom.

#60 [Dokumentacia] Projektová dokumentácia a Dokumentácia riadenia
Zodpovedná osoba: všetci členovia tímu

K prvému kontrolnému bodu je potrebné opísať priebežný stav projektu a opísať jednotlivé moduly riešenia. Rovnako je potrebné opísať priebeh manažovania v našom tíme.

6. Popis projektu (4. - 6. šprint)

Popis projektu v tejto kapitole zahŕňa štvrtý až šiesty šprint. Sú tu opísané zmeny v celkovej architektúre ako i v jednotlivých moduloch, ktoré vznikli po prvom kontrolnom bode. Ku každej časti sú v závere spomenuté úlohy, ktoré sa k jej vypracovaniu viažu. Prehľad jednotlivých úloh je opísaný po šprintoch v ďalších kapitolách.

6.1 Architektúra systému

V tejto časti došlo k malým zmenám v architektúre. V hlavnom programe pribudla podpora viacerých vlákien, takže mnohé moduly môžu bežať paralelne. V rámci tejto novinky samozrejme pribudla správa vlákien a potrebná synchronizácia.

Druhou dôležitou novinkou bola podpora konfigurácie programu a jednotlivých modulov. Počas vývoja vznikla potreba mnohé časti kódu konfigurovať, načítavať konštanty zo súboru a pod. Keďže viaceré moduly potrebovali takúto konfiguráciu, dohodli sme sa, že je to spoločná vlastnosť a má to riešiť architektúra, ktorá by modulom poskytla konfiguračné nastavenia.

V tejto fáze sa naďalej pokračovalo v integrácii modulov, na konci tejto fázy by mal byť prototyp hotový a prepojený so všetkými modulmi. Dôležitou úlohou architekta bola najmä podpora, vysvetlenie prepojenia a pomoc pri generovaní modulov.

6.1.1 Súvisiace úlohy

#66 [Carlos] Integracia modulov
#65 [Architektura] Konfiguracia pre Carlos
#56 [Architektura] Riadiaci modul

6.2 Modul spracovania obrazu

Modul spracovania obrazu vykonáva funkcie z oblasti počítačového videnia, ktoré spracúvajú obraz zo vstupu.

Tento modul v súčasnosti poskytuje nasledovné funkcie:

- *detekcia objektov*
- *kalibrácia kamery*
- *detekcia horizontu*

V tejto kapitole je podrobne opísaná **detekcia horizontu**, ktorá predstavuje novú funkcionalitu tohto modulu. Detekcia horizontu bola vytvorená pre potreby hry *Lietadlo*, ktorá je podrobnejšie opísaná v časti *6.5 Modul rozšírenej reality*. Táto funkcia na vstupnom obraze zdeteguje horizont a nájde tie časti, ktoré zodpovedajú oblohe.

6.2.1 Vstupy

- stream z kamery

6.2.2 Výstupy

- bitmapa veľkosti streamu, kde biele pixely reprezentujú oblohu

6.2.3 Analýza

Jeden z možných spôsobov detekcie horizontu je využitie hranového detektora. Na detekciu hrán sme vyskúšali nasledovné metódy:

- *Canny*
- *Prewitt*
- *Sobel*

Tieto metódy sme vyskúšali aplikovať na obrázkoch, kde je prítomný horizont pri rôznych podmienkach. Najlepšie výsledky dosahoval *Canny detektor*, ktorý bol následne implementovaný do tohto modulu.

6.2.4 Návrh

Na čiernobiely obrázok je najprv aplikovaný Gaussian Blur filter na potlačenie šumu. Potom sa pomocou Canny detektora nájdu hrany na obrázku, čím vznikne binárny obraz s detegovanými hranami. Treshold pri tomto detektore je dostatočne vysoký, aby na oblakoch neboli rozoznané hrany. Na tomto obrázku sa potom nájdu kontúry, pričom algoritmus predpokladá, že nad najvrchnejšou kontúrou sa nácháza obloha.

Bitmapa oblohy sa zostrojí nasledovne. Obraz s kontúrami sa prechádza postupne zhora po stĺpcoch. Algoritmus zafarbuje pixely na bielo, pokým nenájde prvú kontúru v danom stĺpci. Zvyšné pixely sú potom čierne.

Obr.1 Pôvodný obrázok a obrázok s detegovanou oblohou (biela farba)

6.2.5 Implementácia

Táto funkcia bola implementovaná v C++ s knižnicou OpenCV. Funkcia, ktorá príjme vstupný obraz, vráti binárny obrázok so zdetegovanými regiónom oblohy.

V tomto štádiu funkcia predpokladá, že na obrázku je vždy prítomný horizont a obloha.

6.2.6 Testovanie

Na testovanie bolo použité viacero obrázkov s prítomnou oblohou zachytených prevažne pri dobrých svetelných podmienkach. Funkcia bola otestovaná na obrázkoch s jasnou oblohou ako i oblačnou.

6.2.7 Súvisiace úlohy

#51 [OpenCV] detekcia horizontu
#68 [OpenCV] fake funkcionality modulu spracovania obrazu
#69 [OpenCV] funkcia vracajúca polohu oblohy
#70 [OpenCV] úprava funkcionality detekcie objektov do projektu
#81 [OpenCV] finalizácia detekcie horizontu

6.3 Modul Kinect

V module kinectu pokračoval vývoj v smere zisťovania súradníc v reálnom priestore, aby bolo možné odosielať reálne údaje na ďalšie spracovanie. V tejto kapitole je opísaná inicializačná fáza programu, ktorá tvorí hlavnú časť potrebnú pre zisťovanie týchto súradníc.

6.3.1 Vstupy

- RGB video stream z Kinectu
- stream hĺbkových informácií z Kinectu
- 4 nekolineárne body v reálnom priestore (ich súradnice)

6.3.2 Výstupy

- transformačná matica M

6.3.3 Analýza

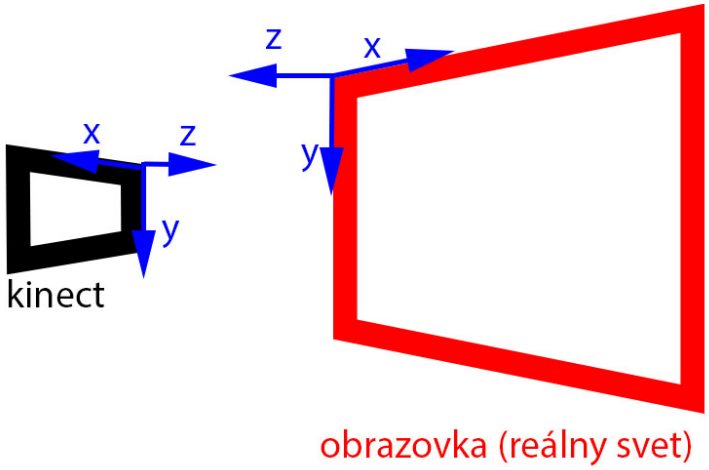
Modul doteraz bol schopný zaznamenať súradnice bodov len z pohľadu kinectu. Pre účely celého systému je ale potreba prispôsobiť tieto súradnice inej súradnicovej sústave, s ktorou už systém pracuje. Preto je potrebné nájsť transformáciu medzi bodmi týchto 2 súradnicových sústav. Na vypočítanie tejto tranformácie je potrebné získať 4 nekolineárne body z každej z týchto sústav. Tieto body usporiadané do matíc vyhovujú modelu transformácie v 3D pristore, ktorého rovnica je:

$$y = M * x$$

kde y je matica súradníc výslednej sústavy, x je matica súradníc východzej sústavy a M je daná transformačná matica. Úpravou tejto rovnice dostaneme rovnicu

$$M = y * x^{-1}$$

v ktorej už poznáme alebo vieme vypočítať všetky členy na pravej strane rovnice.



Obr.1 Náčrt riešeného problému

6.3.4 Návrh

Modul počas inicializácie štyrikrát zaznamená súradnice terča, ktorého stred sa vždy nachádza v bode odpovedajúcom bodu v súradniciach reálneho sveta. Po zaznamenaní všetkých 4 bodov prebehne výpočet transformačnej matice, ktorá sa uloží pre ďalšie použitie v programe.

6.3.5 Implementácia

Pri implepmentácii boli využité funkcie knižnice OpenCV, ktorá ponúka inverziu matíc a tiež násobenie matíc - obe funkcie potrebné pre výpočet transformačnej matice.

Detekcia bodov z pohľadu kinectu je vykonávaná pomocou detekcie terčíka pozostávajúceho z niekoľkých sústredných kruhov. Na detekciu je využitá funkcia HoughCircles z knižnice OpenCV schopná detekovať stred kruhu.

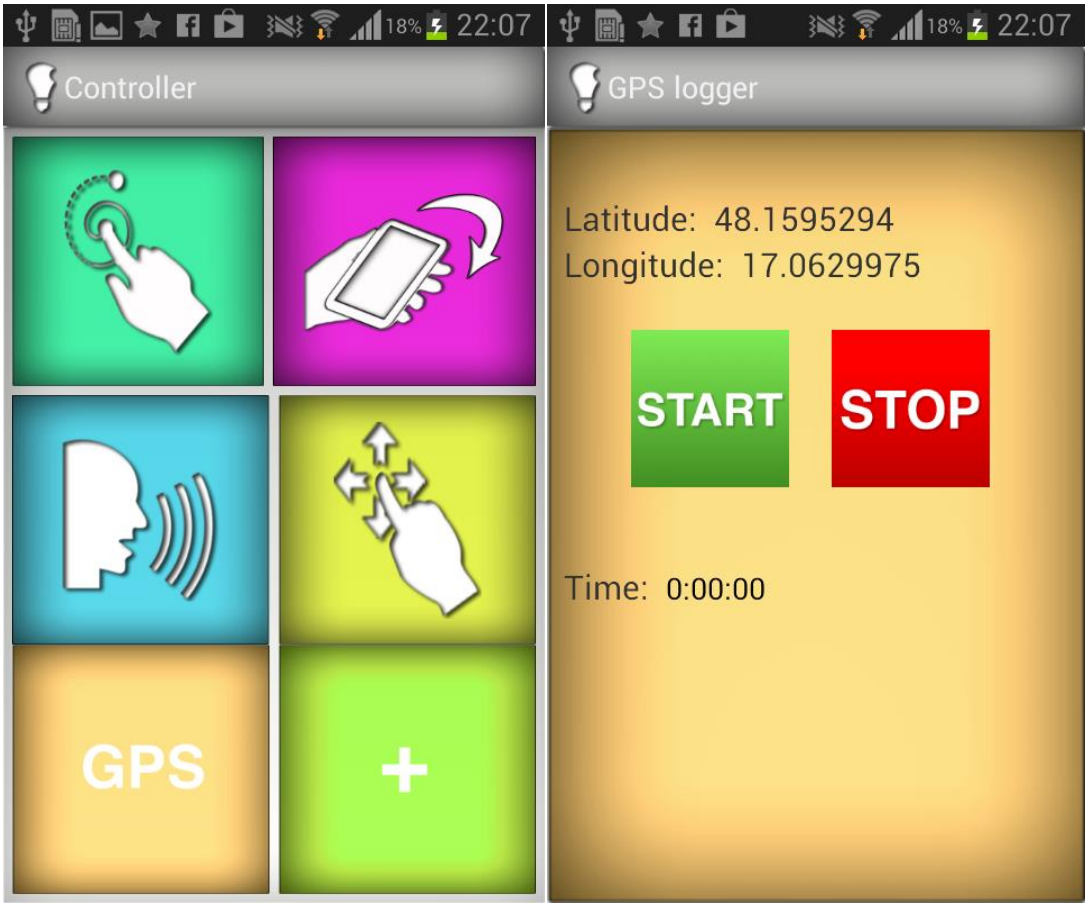
Na ukotvenie bodov v reálnom priestore bol použitý systém nitiek. Tento systém využíva možnosť definovať bod v priestore 3 vektormi a tvoria ho teda pre každý bod 3 nitky, z ktorých každá je uchytená v inom bode. Pre bod so súradnícami [x, y, z] by to boli nitky:
1. uchytená v bode [0, 0, 0] s dĺžkou ako vektor [x, y, z]
2. uchytená v bode [x, 0, 0] s dĺžkou ako vektor [0, y, z]
3. uchytená v bode [0, y, 0] s dĺžkou ako vektor [x, 0, z]

6.3.6 Súvisiace úlohy

#72 [Kinect] Dummy modul
#76 [Kinect] Výpočet transformačnej matice
#89 [Kinect] Detekcia bodov pre inicializáciu
#90 [Kinect] Určenie vstupných bodov pre inicializáciu v reálnom priestore

6.4 Modul Android

V tomto module bolo vykonaných niekoľko menších zmien a taktiež boli do aplikácie doplnené ďalšie funkčné vlastnosti. Jednou z nich bolo začlenenie GPS zaznamenávanie zo samostatnej aplikácie priamo do hlavnej Android časti. S týmto začlenením súvisel aj grafický návrh GPS časti aplikácie tak, aby korešpondovala s grafickým návrhom celej aplikácie. S pridávaním GPS modulu do aplikácie súvisel aj refaktoring pôvodného zdrojového kódu.



Obr.1 Obrazovky Android časti

Druhým doplnkom aplikácie bolo vytvorenie záložného ovládania, ktoré bude slúžiť ako sekundárne ovládanie pre hru. Týmto doplnkom sme chceli hlavne predísť neúspešnému ovládaniu hry z dôvodu zlyhania niektorého z predchadzajúcich možností ovládania - hlasové, dotykové alebo pohybové ovládanie.

Ďalšou zmenou v tomto module bolo prepracovanie serverovej časti z dôvodu nesprávnej funkčnosti. Serverovú časť sa podarilo úspešne implementovať tak, aby posielala správne dáta TCP serveru.

Neočakávanú chybu v časti rozpoznávania hlasu bolo potrebné opraviť a implementovať časť rozpoznávania hlasu inak.

6.4.1 Testovanie

Implementované časti boli otestované úspešne. Špeciálne bolo otestované ovládanie hry pomocou dotykového ovládania, ktoré dopadlo kladne.

6.4.2 Súvisiace úlohy

#78: [Mobil] Prerobit server cast
#73: [Mobil] Pridanie GPS modulu
#74: [Mobil] Prisposobenie vzhladu GPS
#75: [Mobil] Vytvorit zalozne ovladanie
#77: [Mobil] Voice recognition
#71: [Mobil] Refaktoring GPS logger
#83: [Mobil] Pripravit GPS instalacku na web
#84: [Mobil] Otestovanie ovladania hry
#108: [Mobil] vlozenie doxygen komentarov

6.4.3 Komunikácia - Modul Android a Riadiaci modul

Pri klasickom testovaní na statickom mieste android zariadenie posiela rovnakú GPS súradnicu. Keďže testovanie v teréne nie je vždy možné vytvorili sme si GPS súbor s raz nahranými súradnicami. Tieto GPS súradnice sa teda načítavajú zo súboru a táto funkcionalita je implementovaná v TCP serveri, aby čo najviac simulovala posielanie súradnic z androidu. Táto časť sa naimplementovala a otestovala.

Druhou veľkou úlohou bolo konečné prepojenie android zariadenia s tcp serverom a následne preposielanie správ s tcp servera do hlavného modulu. Táto úloha čiastočne zasahuje do celkovej architektúry a aj do modulu hry. Preto sa na tejto úlohe podieľal architekt a aj človek zodpovedný za modul hry. Na začiatku tejto fázy bol TCP server pripravený, dôležitou úlohou bolo definovanie správ, ktoré sa budú posielať a otestovanie posielania všetkých správ.

6.4.3.1 Súvisiace úlohy

#67 [Android] Fake gps pozicia
#57 [Architektura] Spracovanie sprav z Androidu
#105 [Architektura] Prepojenie modulu hry - tcp servera - androidu

6.5 Modul rozšírenej reality

Modul rošírenej reality vykonáva funkcie z oblasti počítačovej grafiky, ktoré vykresľujú upravený obraz zo vstupu.

Tento modul v súčasnosti poskytuje nasledovné funkcie:

- *zobrazenie scény s 3D modelom*
- *zobrazenie videa - prostredia, v ktorom sa nachádza auto*
- *prepínanie medzi stavmi hry - úvodna obrazovka, hranie hry, game over, skóre*
- *implementovanú hernú logiku, zatiaľ bez horizontu*

V tejto kapitole je podrobne opísaná pridaná funkcionalita počas šprintov 4 - 6.

6.5.1 Vstupy

- obrázky z videa
- ovládanie z android časti

6.5.2 Výstupy

- zobrazené video na bočnom skle auta ovládané pomocou mobilu - android

6.5.3 Analýza

Logiku hry Lietadlo je potrebné naimplementovať čo najlepšie, aby sme dosiahli čo najreálnejšie výsledky. Je potrebné brať do úvahy fakt, že lietadlo má určité rozmery a stačí ak sa akoukoľvek časťou lietadla dotkne okraja resp. horizontu. Následne, by hra mala skončiť.

Celú hru Lietadlo je potrebné rozkúskovať do viacerých stavov a zamyslieť sa aj nad spôsobom reštartu.

6.5.4 Návrh

Ako najvhodnejšie riešenie logiky hry je celé lietadlo zabaliť do kvádra a zisťovať či sa tento kváder dotýka pozadia. Ak áno, tak lietadlo je na hernom pláne. Ak nie, tak hra končí, lebo vyletelo mimo obraz. Obdobne je to potrebné implementovať aj pre horizont.

Navrhované stavy sú:

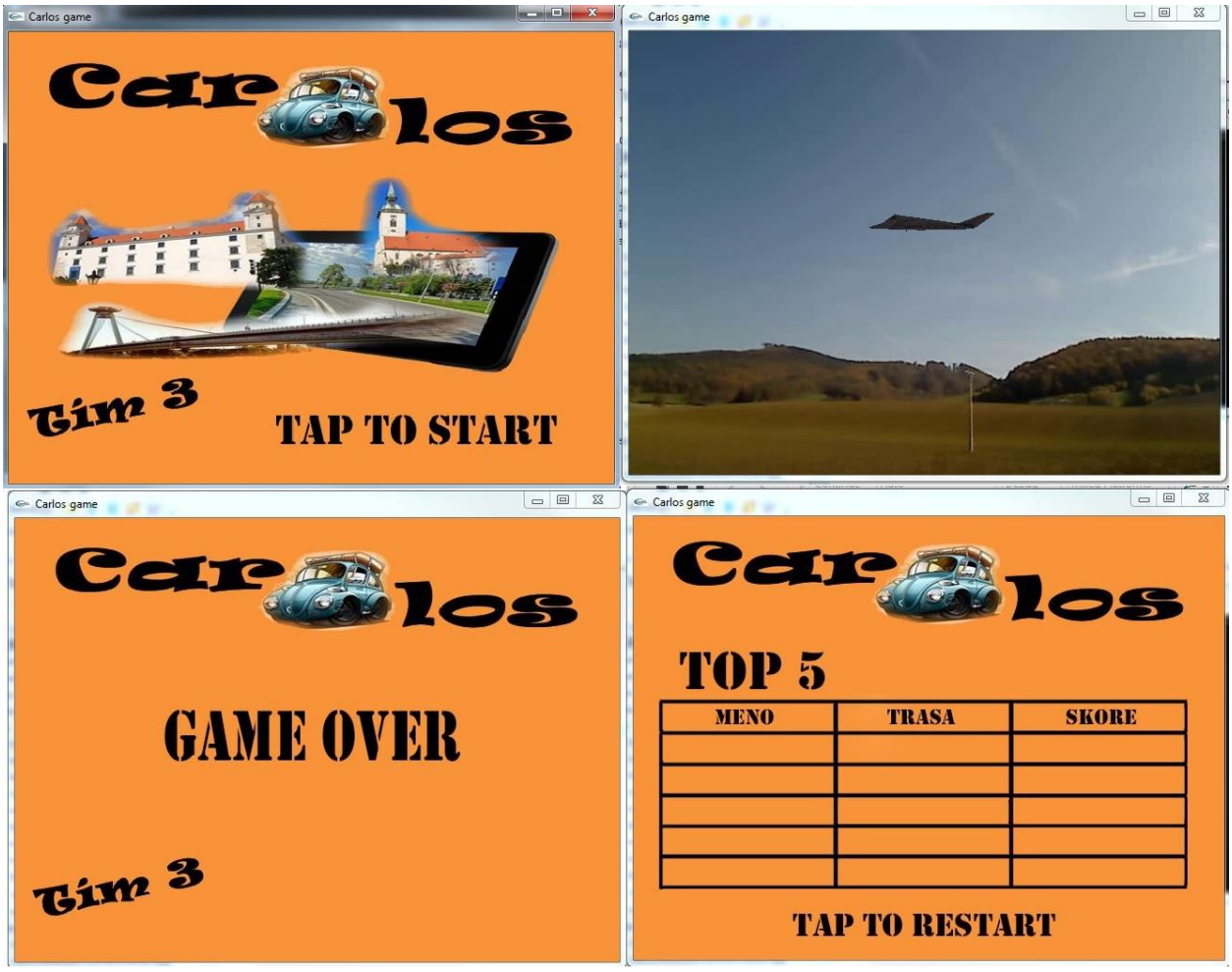
- úvodná obrazovka
- hranie hry
- game over
- obrazovka so skóre

Taktiež je potrebné upraviť spôsob ovládania, prepojiť sa s Lukášovou časťou a začať prijímať ovládanie z jeho modulu.

6.5.5 Implementácia

V tomto momente už je naimplementovaná počiatočná logika hry, ktorá zisťuje, kde sa lietadlo nachádza. Sú vytvorené všetky navrhnuté stavy zo sekcie 6.6.3. Hru je už možné hrať, aj keď nie je zobrazený horizont a nepočíta sa ani skóre.

Ovládanie sa už prijíma z hlavného modulu, ktorý je opísaný v sekcii 2.1.4.



Obr.2 Jednotlivé stavy hry - Lietadlo. Vľavo hore - úvodná obrazovka, vpravo hore - hranie hry, vľavo dole - game over a vpravo dole - obrazovka skóre.

6.5.6 Testovanie

Na testovanie bolo použité video, ktoré sme si nahrali. Otestovali sme aj jednotlivé stavy hry, prijímanie obrázkov a príkazov. Tieto príkazy sa používajú na ovládanie lietadla. Testovanie dopadlo podľa očakávaní a prípadne problémy boli odstránené.

6.5.7 Súvisiace úlohy

#61 [OpenGL] Dokoncit zobrazenie videa
#63 [OpenGL] Vytvorenie hernej logiky
#64 [OpenGL] Zobrazenie textu v hre
#103 [OpenGL] Napojenie na git, gerrit a doxygen komentare, spojenie s lukasovou castou
#104 [OpenGL] Opravenie spojenia s lukasovou castou
#86 [OpenGL] Vlozenie doxygen komentarov
#87 [OpenGL] Nastavenie kniznic
#88 [OpenGL] Prepojit sa s robom - ovladanie pomocou mobilu

6.6 Modul databázy

V module databázy bola doimplenetovaná databázova služba, ktorá pokrýva všetku prácu nad databázou. Databáza bola naplnená potrebnými údajmi. Tieto nové časti boli riadne otestované a pre prvú verziu prototypu je táto časť hotová.

6.6.1 Súvisiace úlohy

#53 [Databaza] OOP Mapovac a DB Service

7. Štvrtý šprint

Začiatok šprintu: 18.11.2013

#56 [Architektúra] Riadiaci modul
Zodpovedná osoba: Lukáš Sekerák

Náš hlavný program beží pod určitým operačným systémom. Keďže aplikácia beží neustále, bolo potrebné naimplementovať cyklus, v ktorom sa aplikácia nezastaví, teda len za určitých podmienok. Zároveň v rámci tohto cyklu musí aplikácia pravidelne komunikovať s oper. systémom. V tejto úlohe sa riešili vyššie spomenuté problémy.

#57 [Architektúra] Spracovanie správ z Androidu
Zodpovedná osoba: Lukáš Sekerák

Mnohé správy, ktoré sa príjmajú z android zariadenia majú textovú formu. Zvyšok modulov však používa správy definované v ENUM. Preto je v tejto úlohe potrebné tieto správy prerobiť. Zároveň takáto komunikácia bude prebiehať cez viacero vlákien a túto operáciu treba synchronizovať.

#61 [OpenGL] Dokončiť zobrazenie videa
Zodpovedná osoba: Peter Hamar

Na bočné sklo auta je potrebné zobraziť to, čo je práve vidieť za oknom. Preto musíme v hre zobrazovať na pozadí video. Video musíme rozkúskovať do obrázkov a tie následne zobrazovať. Problém bol vyriešený iba sčasti, keďže pre správny beh boli potrebné vlákna. Dokončené v nasledujúcom šprinte.

#63 [OpenGL] Vytvorenie hernej logiky
Zodpovedná osoba: Peter Hamar

Bolo potrebné začať vytvárať hernú logiku a určovať pozíciu lietadla v rámci obrazovky. Taktiež sme v tomto bode rozdelili hru do viacerých stavov a dané stavy na seba vzájomne poprepájali. Týmto sme dosiahli to, že hru už je možné hrať.

#64 [OpenGL] Zobrazenie textu v hre
Zodpovedná osoba: Peter Hamar

Keďže sme si definovali do akých stavov sa môže hra dostať, bolo potrebné si pripraviť jednotlivé textúry - nakresliť ich. Vytvoriť spôsob reštartu hry a taktiež sa zamerať na to, aby bolo možné jednoduchým spôsobom zobraziť text v okne.

#65 [Architektúra] Konfigurácia pre Carlos
Zodpovedná osoba: Lukáš Sekerák

Počas vývoja vznikla potreba mnohé časti kódu konfigurovať, načítavať konštanty zo súboru a pod. Keďže viaceré moduly potrebovali takúto konfiguráciu, dohodli sme sa, že je to spoločná vlastnosť, ktorú má riešiť architektúra, ktorá by modulom poskytla konfiguračné nastavenia. V tejto úlohe sa riešila konfigurácia.

#66 [Carlos] Integrácia modulov
Zodpovedná osoba: Lukáš Sekerák

V tejto fáze sa naďalej pokračovalo v integrácii modulov. Dôležitou úlohou architekta je v tejto úlohe najmä podpora, vysvetlenie prepojenia a pomoc pri generovaní modulov.

#67 [Android] Fake gps pozícia
Zodpovedná osoba: Lukáš Sekerák

Pri klasickom testovaní na statickom mieste android zariadenie posiela rovnakú GPS súradnicu. Keďže testovanie v teréne nie je vždy možné, vytvorili sme si GPS súbor s raz nahranými súradnicami. Tieto GPS súradnice sa teda majú načítavať zo súboru a táto funkcionalita má byť naimplementovaná v TCP servery, aby čo najviac simulovala posielanie súradnic z androidu.

#68 [OpenCV] fake funkcionality modulu spracovania obrazu
Zodpovedná osoba: Patrik Polatsek

Cieľom tejto úlohy je definovať formát funkcií, ktoré budú volané z riadiacej časti. Ďalej je potrebné vytvoriť fake verzie týchto funkcií.

#69 [OpenCV] funkcia vracajúca polohu oblohy
Zodpovedná osoba: Patrik Polatsek

V úlohe #51 sa vybral na detekciu horizontu Canny hranový detektor. V tejto úlohe je potrebné vytvoriť reálnu funkciu, ktorá vráti polohu oblohy pre potrebu hry “Lietadlo”.

#70 [OpenCV] úprava funkcionality detekcie objektov do projektu
Zodpovedná osoba: Patrik Polatsek

Vytvorenú funkciu detekcie objektov v rámci modulu spracovania obrazu je potrebné zakomponovať do spoločného solution.

#103 [OpenGL] Napojenie na git, gerrit a doxygen komentáre, spojenie s Lukášovou časťou
Zodpovedná osoba: Peter Hamar

Cieľom tejto úlohy bolo začať používať git a gerrit. Zdrojový kód je potrebné komentovať pomocou nástroja doxygen, čiže bolo potrebné povkladať komentáre do kódu. Pre zobrazenie videa bolo potrebné sa prepojiť s hlavnou (Lukášovou) časťou. Toto prepojenie nefungovalo úplne správne a bude potrebná oprava v naslednujúcom šprinte.

#95 [Support] Nainštalovať programy na pc v škole
Zodpovedná osoba: Martin Petluš

Na novo zakúpený pc bolo treba nainštalovať vývojové prostredia pre C++/C, Javu. Tiež bolo treba nainštalovať a nastaviť OpenCV ako aj všetky ostatné programy nutné pre základnú prácu s pc.

#94 [Support] Dokončenie Gerritu
Zodpovedná osoba: Martin Petluš

Dokončiť inštaláciu Gerritu na serveri. Otestovať jeho funkčnosť a prihlásenie nových používateľov. Napísať návod na jeho použitie ostatnými členmi tímu.

8. Piaty šprint

Začiatok šprintu: 28.11.2013

#71 [Mobil] Refaktoring GPS logger
Zodpovedná osoba: Róbert Sabol

Pri začlenovaní GPS aplikácie do hlavnej Android časti sa našli isté nedostatky v pomenovaní premenných, preto bolo potrebné kód zrefaktorovať.

#72 [Kinect] Dummy modul
Zodpovedná osoba: Jakub Mercz

Pre možnosť fungovania systému aj bez potreby fyzického zariadenia kinectu a jeho konfigurácie, bol vytvorený dummy modul odosielajúci falošné informácie.

#73 [Mobil] Pridanie GPS modulu
Zodpovedná osoba: Róbert Sabol

Tažisko tejto úlohy bolo hlavne v tom, aby samostatne vytvorená GPS aplikácia bola zahrnutá v hlavnej Android aplikácii Controller. Bolo potrebné vytvoriť novú aktivitu v aplikácii ako aj xml súbory na layout a pod.

#74 [Mobil] Prispôsobenie vzhladu GPS
Zodpovedná osoba: Róbert Sabol

Keďže grafika aplikácie je navrhnutá tak, aby všetky aktivity boli podobne graficky spracovane, bolo potrebné pridanú GPS časť graficky upraviť tak, aby korešpondovala ku grafike celej aplikácie.

#75 [Mobil] Vytvoriť záložné ovládanie
Zodpovedná osoba: Róbert Sabol

Aj keď ovládanie pomocou pohybových senzorov telefónu alebo pomocu rôznych dotykových gest, je veľmi efektné’pre koncového používateľa, pre eliminovanie rizika zlyhania ovládania pomocou týchto senzorov, sme boli nútení vytvoriť záložné ovladanie, ktoré obsahuje len tlačidlá s danými akciami.

#76 [Kinect] Výpočet transformačnej matice
Zodpovedná osoba: Jakub Mercz

Implementácia a anlýza možností transformácie medzi 2 súradnícovými sústavami, ktorej výsledkom bola transformačná matica M

#77 [Mobil] Voice recognition
Zodpovedná osoba: Róbert Sabol

Pri testovaní hlasových povelov nastal problém pri rozpoznávaní príkazov preto bolo potrebné ošetriť isté chybové stavy.

#78 [Mobil] Prerobiť server časť
Zodpovedná osoba: Róbert Sabol

Dodaná serverová časť v Android aplikácii nevyhovovala naším potrebám, keďže pracovala s JSON-mi, preto sa úplne prerobila na TCP komunikáciu, ktorá korektne dokázala komunikovať s C++ serverom.

#81 [OpenCV] finalizácia detekcie horizontu
Zodpovedná osoba: Patrik Polatsek

Na základe poznatkov získaných pri úlohe #51 a #69 dokončiť funkciu, ktorá bude na vstupnom obraze hľadať polohu oblohy.

#82 [OpenCV] začlenenie modulu spracovania obrazu do projektu
Zodpovedná osoba: Patrik Polatsek

Začleniť všetky 3 funkcie modulu spracovania obrazu do spoločného solution.

#96 [OpenCV] Git - použiť systémové premenné namiesto priamych ciest (OpenCV)
Zodpovedná osoba: Martin Petluš

Odskúšať, či sa dajú použiť v projekte Visual Studia systémové premenné ku knižnici OpenCV namiesto absolútnej cesty k OpenCV.

#104 [OpenGL] Opravenie spojenia s Lukášovou časťou
Zodpovedná osoba: Peter Hamar

Je potrebná oprava spojenia medzi modulom hry a hlavnou časťou. Nefungovali správne 2 okná súčasne - textúry.

9. Šiesty šprint

Začiatok šprintu: 5.12.2013

#83 [Mobil] Pripravit GPS inštalačku na web
Zodpovedná osoba: Róbert Sabol

Kvôli potrebám natáčania testovacieho videa bolo potrebné aby GPS logger fungoval ako samostatná aplikácia. Preto bol modul GPS vyňatý z Controller-a a skompilovaný do samostatnej APK aplikácie, ktorá sa dá jednoducho nainštalovať.

#84 [Mobil] Otestovanie ovládania hry
Zodpovedná osoba: Róbert Sabol

Po sfunkčnení hry bolo potrebné otestovať, či ovládanie pomocou Android aplikácie bude dostatočne dobré a rýchle pre naše potreby. Android aplikáciu sme s hrou spojili TCP serverom tak, že aplikácia vytvorila spojenie so serverom a odosielala TCP sokety serveru, ktorý ich následne odosielal modulu s hrou.

#86 [OpenGL] vloženie doxygen komentárov
Zodpovedná osoba: Peter Hamar

Po rozšírení a úprave kódu bolo potrebé opraviť a dopísať doxygen komentáre.

#87 [OpenGL] Nastavenie knižníc
Zodpovedná osoba: Peter Hamar

Keďže modul hry - Lietadlo používa vlastné knižnice a chceme mať projekt uložený na gite, aby si ho vedeli spustiť aj ostatní členovia tímu, je potrebné v projekte použiť systémové premenné. Taktiež uploadnuť tieto knižnice na google drive pre ostatných členov tímu a dať im zoznam systémových premenných ktoré potrebujú.

#88 [OpenGL] prepojiť sa s Robom - ovládanie pomocou mobilu
Zodpovedná osoba: Peter Hamar

Doposiaľ bolo ovládanie riešené pomocou klávesnice. Teraz je potrebné sa prepojiť s časťou android, cez hlavný - riadiaci modul - ovládanie už bude posielané spolu s obrázkom.

#89 [Kinect] Detekcia bodov pre inicializáciu
Zodpovedná osoba: Jakub Mercz

Detekcia bodov pomocou kinectu pre účely transformačnej matice.

#90 [Kinect] Určenie vstupných bodov pre inicializáciu v reálnom priestore
Zodpovedná osoba: Jakub Mercz

Určenie bodov v reálnom priestore pre úcely transformačnej matice

#91 [Support] Git spraviť na PC v škole - rozbehať projekt
Zodpovedná osoba: Martin Petluš

Nastaviť Git v škole na náš projekt. Nastaviť všetky systémové premenné. Otestovať, či sa dá projekt spustiť.

#97 [OpenCV] opraviť modul spracovania obrazu
Zodpovedná osoba: Patrik Polatsek

Pri testovaní funkcie detekcie objektov bolo objavených niekoľko chýb. Cieľom tejto úlohy bolo zistené úlohy opraviť.

#98 [OpenCV] opraviť chybu s heap vo vs 2012
Zodpovedná osoba: Patrik Polatsek

Po presune modulu spracovania obrazu do spoločného solution bola objavená chyba OpenCV vo vs 2012 na 64bit systémoch pri vykonávaní niektorých funkcií akou je napríklad detekcia keypointov a ich opis do matice deskriptorov a označená ako RtlFreeHeap error. Je potrebné zabezpečiť kompatibilitu spomínaných funkcií aj po presune modulu z vs 2010 na vs 2012.

#101 [OpenCV] vloženie doxygen komentárov
Zodpovedná osoba: Patrik Polatsek

Celý projekt modulu spracovania obrazu je potrebné okomentovať doxygen značkami.

#102 [Dokumentacia] 2. kontrolny bod
Zodpovedná osoba: všetci členovia tímu

K druhému kontrolnému bodu je potrebné znovu odovzdať Projektovú dokumentáciu a Dokumentáciu riadenia, kde treba uviesť nové informácie a zmeny oproti prvému kontrolnému bodu, ktoré boli zadokumentované v rámci úlohy #60.

#105 [Architektura] Prepojenie modulu hry - tcp servera - androidu
Zodpovedná osoba: Lukáš Sekerák

V rámci dokončenia prepojenia modulu hry - tcp servera - androidu je potrebné dodefinovať správy, ktoré sa budú posielať. Tieto správy poslať z android zariadenia do tcp servera a odtiaľ do modulu hry. Všetko je potrebné riadne otestovať a zabezpečiť.

#105 [Architektura] Doxygen komentare
Zodpovedná osoba: Lukáš Sekerák

Niektoré časti zdrojového kódu nie sú riadne okomentované. Je potrebné doplniť anotáciu pre doxygen. Komentáre, ktoré existujú treba skontrolovať, či spĺňajú anotáciu.

#107 [TextPosModule] Vloženie Doxygen komentárov
Zodpovedná osoba: Martin Petluš

Okomentovať zdrojové kódy v module Doxygen značkami.

#108: [Mobil] vlozenie doxygen komentarov
Zodpovedná osoba: Róbert Sabol

Celý projekt modulu Android bolo potrebné okomentovať doxygen značkami.