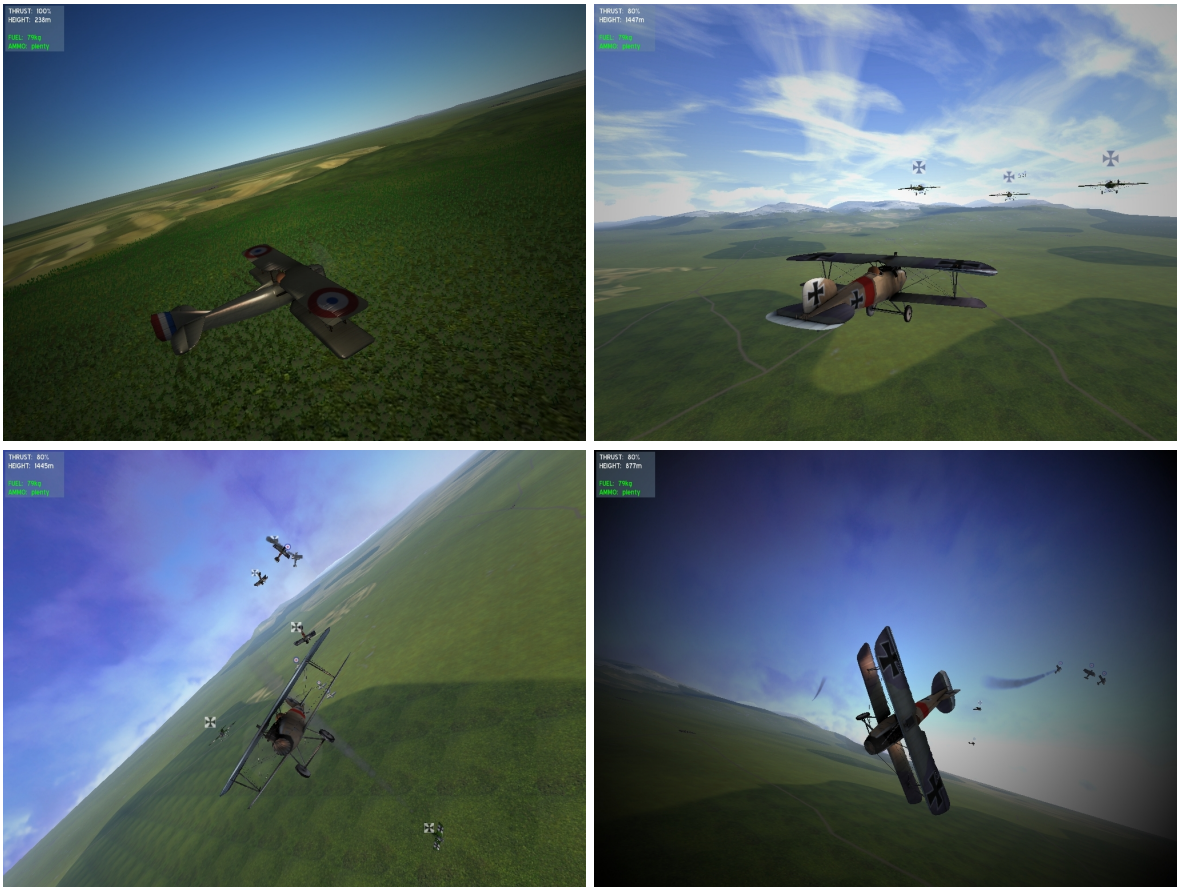
# Postrehy k SW projektu - Flying Samurai

Oskár Elek*

16. srpna 2010



**Úvod** Účelom tohto dokumentu je zachytiť niektoré hlavné rysy riešenia softwarového projektu na MFF vrámci NMgr. oboru Softwarové Systémy. Je dosť pravdepodobné, že časť toho, čo tu zaznie, sa už objavila v starších a možno už aj novších obdobných dokumentoch od iných tímov. Je možné, že čítať veľmi podobné myšlienky stále dokola môže byť unavujúce; na druhej strane je potreba vziať do úvahy, že vzhľadom k tomu, koľko času vám zaberie vypracovanie projektu, je dôsledná príprava nanajvýš dôležitá a čítanie zamyslení ako je toto vám pomôže sa vyvarovať chýb, ktoré urobili tí pred vami. Naviac — ak sa nejaký názor často opakuje, svedčí to len o tom, že je naozaj relevantný. Nenechajte sa odradiť dĺžkou tohto dokumentu — snažil som sa, aby bol čo najdetailnejší a najobecnejší.

*oskee@centrum.cz

**Tím** Alebo inak povedané ľudia, ktorí na projekte pracujú. Toto je prvá a asi aj najdôležitejšia vec, ktorú budete pri vznikajúcom projekte riešiť — s tímom projekt stojí a padá. Sú dve kľúčové otázky, ktoré je potreba zodpovedať: „kto?“ a „koľko?“.

Vybrať tých správnych ľudí na projekt nie je vôbec jednoduché. Matfyzáci sú rôznorodí ľudia a nájsť skupinku, ktorá si sadne, je dosť syzifovská úloha. Naviac na Matfyze koluje fáma, že nie je dobrý nápad, aby v tíme boli ľudia, ktorí sú priatelia, pretože sa ich kamarátstvo môže ľahko skončiť. S tým však nesúhlasím. V našom prípade to bolo presne naopak — s ľuďmi, s ktorými som na projekte pracoval, sme sa spoznali ešte lepšie, jednoducho preto, že nad projektom nejaký ten čas spolu strávite. Obecne to veľmi závisí na ľuďoch samotných, ako sú schopní zvládať stres a podobne. Ak však vidíte, že si s ľuďmi, s ktorými hodláte započať projekt, moc nesedíte už vopred, tak je isté, že v priebehu projektu sa to len zhorší. Hlavná personálna požiadavka na členov tímu je schopnosť komunikovať — nie je nič otravnejšie, ako keď jeden člen tímu na porade už polhodinu vysvetľuje nejakú banalitu, pričom je treba prebrať ešte ďalších päť vecí (alebo sa naopak vôbec nevie vyjadriť). Toto však bude na Matfyze žiaľ pomerne častý problém.

Dôležité sú aj technické znalosti. To by na Matfyze u ľudí, ktorí preliezli aspoň toho bakalára, nemal byť problém. Stať sa ale môže čokoľvek, a preto je dobré, v prípade že daného človeka nepoznáte — zistite, na akých osobných projektoch robil predtým (sú ľudia, ktorí zašli tak ďaleko, že chceli potenciálnym členom tímu kontrolovať index, ale to je podľa mňa zbytočný, a možno aj dosť detinský, extrém). Nutné minimum je, aby všetci rozumeli zvolenej platforme a technológiám, ktoré budú pri riešení použité, aby sa už neskôr nestrácal čas nejakým rozsiahlym štúdiom materiálov (od toho je príprava).

Počet ľudí je podľa súčasných podmienok (ak. rok 2009/2010) stanovený na 4–8. Ideálne číslo sa javí ako 5; 6 a viac ľudí sa už ťažko koordinuje (nájdenie spoločného času na pravidelné schôdze, rozdeľovanie úloh tak, aby bolo čo najmenej závislostí na vzájomnej práci atp.) a 4 už je zase trochu málo (ak jeden načas vypadne — a to sa pravdepodobne stane, Murphyho zákony platia aj na MFF — zostanú dočasne len traja a to už je naozaj málo). My sme začali piati, bohužiaľ sme sa z určitých dôvodov museli s jedným z členov tímu ešte pred zahájením rozlúčiť, a miestami bolo cítiť, že štyria ľudia sú málo, obzvlášť na počítačovú hru (viď nasledujúci odstavec).

Nakoniec, je veľmi vhodné, aby sa jeden z členov tímu ujal akéhosi vedenia. Projekt ma síce vedúceho z radov vyučujúcich (prípadne externistu, ako to bolo u nás), ale jeho úlohou je dohliadať len na celkové smerovanie projektu. Zmienený vybraný člen tímu bude naopak mať hlavné slovo pri mikromanažmente — organizovanie schôdzí, administratívne veci ohľadom projektu, buzerovanie flákajúcich sa členov tímu atp. :) Je to trochu nevďačná úloha, ale zase sa pri tom človek niečo aj naučí a vrámci tímu bude mať hlavné slovo práve on (prípadne ona). Tento človek na to ale musí mať vlohy — najčastejšie to bude osoba typu „buldozér“ (vysvetlenie snáď nie je potreba). U nás sme sa neštandardne rozhodli, že takíto koordinátori budeme hneď dvaja — čo sa ukázalo ako dobrý nápad, pretože aspoň všetka táto extra zodpovednosť neležala na jedinom človeku. Ďalšia výhoda tohto prístupu je v tom, že ak jeden z týchto dvoch ľudí z nejakého dôvodu dočasne vypadne, ten druhý ho bez problémov zastúpi (čo sa stalo neraz).

Ak už je teda tím zostavený, je načase sa rozhodnúť, čo vlastne bude témou projektu.

**Tematika** Výber témy samozrejme závisí na ľuďoch a ich znalostiach a záujmoch. Tému je možné si vymyslieť vlastnú (tú potom musí komisia odsúhlasiť), alebo sa prihlásiť k nejakému už vypísanému projektu. Výhody a nevýhody oboch prístupov si snáď každý domyslí sám, snáď len: ak chcete mať pri vypracovaní väčšiu voľnosť, vymyslite si vlastnú tému, a naopak.

Čo sa týka rozsahu — od ak. roku 2007/2008 (myslím) je zavedený 9-mesačný limit na vypracovanie projektu (predtým projekty trvali bežne 2–3 roky a ľudia kvôli tomu príliš naťahovali štúdium). Takže, ak si vymýšľate vlastnú tému, tak si najskôr dobre rozmyslite, čo všetko by ste tam za tých 9 mesiacov chceli naimplementovať (pozor, jedná sa o 9 mesiacov hrubého času od zahájenia po obhajobu, a to, že sú medzitým skúškové obdobia je už každého osobný problém). Potom vezmite ten zoznam featur, a tak 40% z nich podľa priority vyškrtajte. Z toho, čo zostane, máte veľmi dobrú šancu dokončiť tak dve tretiny...

V krátkosti by bolo dobré zmieniť tému nášho projektu, pretože každé softwarové dielo je odlišné, takže na pochopenie niektorých našich krokov je dobré vedieť aj špecifiká tohto projektu. Náš projekt

— Flying Samurai — je hra, presnejšie 3D letecký bojový simulátor v reálnom čase z obdobia 1. Svetovej vojny. Obecne sa dá povedať, že hry, najmä tie v 3D, sú aplikácie pomerne náročné na rozpočet, ako aj na technickú realizáciu. Vyplýva to z nutnosti vytvoriť dobre optimalizovaný herný engine, a taktiež vytvoriť rozsiahly virtuálny svet, v ktorom sa hráč bude pohybovať. Keďže rozpočet na SW projekt predstavuje v šúčasnosti presne 0 (nula) KČ, budete si musieť pomôcť, ako to bude možné (a to sa samozrejme netýka len hier). Tým sa pozvoľna dostávame k technickému zázemiu.

**Technológie** U technológií všeobecne platí, že je dobré, aby sa tím zhodol na čo najväčšom množstve vecí. Samozrejmosťou je zhodnúť sa na použitom jazyku (v našom prípade to bolo prirodzene C++), ale je tiež dobré, aby celý tím používal podľa možností rovnaký OS, rovnaké vývojové IDE atď. Výhodou opačného prístupu by bolo, že by sa na rôznych platformách podarilo prísť na väčšie množstvo chýb v projekte, ale je na uvážení, či to stojí za zvýšené množstvo času vynaloženého na nastavovanie prostredia a administráciu. V našom prípade sme predpokladali kompatibilitu hry tak s MS Windows, ako aj s Linuxom (žiadna použitá technológia tomu nebránila), takže namiesto vytvorenia fixného projektu pre Visual Studio sme použili cmake, čo stálo o poznanie viac času. Nakoniec sme snahu o kompatibilitu s Linuxom opustili, a to hlavne kvôli problémom s OpenGL u grafického enginu, ktorý sme použili (Ogre3D). Tiež sme všetci nepoužívali rovnakú verziu Visual Studia, ale kombináciu verzií 2005, 2008 a 2008 Express, čo nám tiež pridalo trochu práce (ale vďaka spomínanému cmake-u zas nie tak veľa). Nakoniec sme všetci plynule prešli na verziu 2008/2008 Express, pretože verzie 2005 a 2008 používajú rôzny formát napríklad .lib knižníc a tak sme predišli dvojitej kompilácií všetkých 3rd party knižníc pri každej zmene ich verzie.

Samotné knižnice z tretích strán sú tiež častou otázkou. Problém je tu jasný — ich použitie môže ušetriť veľa času, ak nájdete to, čo potrebujete, na druhej strane ale môžu spôsobiť množstvo problémov, hlavne kvôli nekvalitnej (alebo úplne chýbajúcej) dokumentácii a chybám, ktoré môžu obsahovať. To sa týka hlavne open-source zdrojov, na ktoré budete s nulovým rozpočtom väčšinu času odkázaní. Aj keď mantra, ktorá sa v súvislosti s open-source projektmi typicky recituje („máte zdrojáky, tak si to opravte/modifikujte/doplňte“) je pekná, v praxi narazíte na problém, že na to jednoducho nemáte čas (obzvlášť pri úrovni dokumentácie, ktorou sú open-source projekty väčšinou vybavené — v priemere sa stretnete s dokumentáciou, ktorá by sa dala označiť ako „skeletálna“). Príklad — na renderovanie 3D terénu sme použili plugin do nami použitého grafického enginu Ogre3D, nazvaný MLP. Po chvíli používania (ktoré tiež nebolo úplne hladké) sme zistili, že sa v tomto plugine nachádza obrovský memory leak, ktorý v zásade vôbec nedealokoval nepotrebné stránky terénu, čo zväčša viedlo k pádu hry po nejakých 15 minútach pri spotrebe RAM cca. 1.5GB. Odstránenie tohto problému nebolo zďaleka jednoduché, zabralo jednému človeku nejaký týždeň čistej práce, pretože nájsť v 1.2MB nezdokumentovaného C++ kódu niečo konkrétne nie je vôbec jednoduché (asi nemusím zdôrazňovať, že sa zďaleka nejednalo o nejaký jeden chýbajúci delete kdesi v deštruktore). Na druhej strane, napísanie vlastného systému pre renderovanie terénu daného rozsahu by vydalo jednému človeku na celý SW projekt...

My sme sa rozhodli pre cestu použiť čo najväčšie množstvo 3rd party knižníc, aj za cenu zmienených rizík. Osobne si myslím, že sa to vyplatilo, pretože sme boli schopní dostať do projektu veci, ktoré by sme bez toho nemali šancu stihnúť. Pre záujemcov nami použité knižnice a frameworky vymenujem konkrétne:

- grafický engine Ogre3D (pomerne spoľahlivý a zavedený projekt s veľkou komunitou, slušným designom a množstvom pluginov, na niekoho vkus môže byť príliš veľký a obecný). Ogre sám ďalej používa množstvo knižníc, hlavne DirectX a OpenGL na renderovanie (my sme použili DX renderer, pretože s OGL rendererom bolo pomerne dosť problémov)

- plugin do Ogre3D na renderovanie terénu MLP (v dobe nášho vývoja jediný použiteľný systém tohto druhu pre Ogre3D, dnes by som kvôli jeho neodladenosti siahol po novom core systéme pre renderovanie terénu, ktorú Ogre3D dostal do vienka vo verzií 1.7)

- plugin do Ogre3D na renderovanie instancovanej geometrie PagedGeometry (výborne zdokumentovaný a stabilný systém, pomerne slušne konfigurovateľný a dobre optimalizovaný)

- systém na renderovanie užívateľského rozhrania CEGUI (príliš nabobtnaný a nemotorný systém, nedoporučujem)
- fyzikálna knižnica Bullet (slušný systém, ale s niektorými funkciami ako CCD boli problémy, stálo by za to preskúmať alternatívy)
- systém na navigáciu virtuálnych agentov OpenSteer (bezproblémový a zavedený systém, bez väčších výhrad)
- knižnica OIS na spracovanie vstupu z periférií (dobre použiteľná jednoduchá knižnica, okrem myši a klávesnice podporuje aj joystick, gamepad atď.)
- zvukový engine irrKlang (zvuk sme robili narýchlo pred koncom, takže neviem veľmi posúdiť, snáď okrem nenáročného použitia)
- XML reader/parser tinyXML (známa knižnička, dobrá na konfiguráky)
- niektoré moduly knižnice boost (netreba predstavovať)

Okrem toho sme použili celú radu softwaru, hlavne na prácu s najrôznejšími grafickými resourcami a dátami (napríklad pre herný terén).

Je teda vidieť, že s určitou dávkou trpezlivosti sa dá uniesť aj väčšie množstvo 3rd party knižníc. Je hlavne dôležité zabezpečiť, aby sa použité technológie prekrývali a ovplyvňovali čo najmenej, ak nie sú priamo navrhnuté na vzájomnú spoluprácu (čo väčšinou nie sú; výnimkou sú napríklad Ogre3D a Ogre ODE). Pri výbere technológií bez ich predchádzajúcej znalosti je to trochu brodenie sa močiarom, určité vodítka o kvalite danej knižnice však môžu predstavovať napríklad projektové fóra (veľkosť užívateľskej základne a typy problémov, ktoré sa na fórach riešia) a dostupná dokumentácia.

Nakoniec ešte zmienim pomocné technológie. Slovom „pomocné“ nemyslím, že by mali nejaký podružný význam, ale skôr to, že ako také nie sú priamou súčasťou projektu. Ich význam je naopak veľký a v prvom rade je ním uľahčenie či urýchlenie vašej práce. Podobne ako u knižníc tretích strán sme používali celkom širokú škálu pomocných technológií:

- Prvá vec, o ktorej by softwarový inžinier nemal ani na moment zaváhať, je repository (počul som hrôzostrašné historky o tom, ako si členovia tímu posielali zdrojáky mailom, ale rozhodol som sa im neveriť). My sme použili Subversion, konkrétne klienta TortoiseSVN, ktorý má celkom priateľské rozhranie a je vôbec nejak osvedčený (minimálne na MFF). SVN nie je dokonalé, má svoje muchy, ale časom sa k nemu naučíte správať slušne a potom to už pôjde bez problémov. Rozhodne sa však malá časová investícia do neho dlhodobo vyplatí.
- Ďalšia dôležitá vec je systém na bugtracking (zaznamenávanie bugov a nových taskov a ich momentálneho stavu — výborná vec, aby ste nemuseli tieto veci hľadať v maili, alebo nebodaj pamätať). Existuje niekoľko alternatív, my sme začali s Bugzillou, ale kvôli jej zvláštnemu rozhraniu sme neskôr plynule prešli na Trac, s ktorým sme boli plne spokojní. Trac vám ponúkne jednoduchú administráciu, pokročilé možnosti triedenia a vyhľadávania, plus všetky ostatné veci, ktoré by ste od takého systému čakali, a naviac je maximálne prehľadný.
- V závislosti na type projektu sa môže viac alebo menej hodiť projektová Wiki. Sem je dobré dávať texty týkajúce sa projektu, či už návody („Ako zbuildit projekt a závislosti pod IDE XYZ“, „Ako sa dostať do repository“ a podobne), články (niekto napríklad preskúma nejakú technológiu a napíše o nej rešerš) a iné zdroje, ktoré je dobré mať na centrálne dostupnom mieste. My sme používali DocuWiki (tuším). Výhoda Wiki je aj to, že sa pomerne jednoducho edituje (trochu podobne, ako LaTeX) a vyzerá dobre.
- Vzhľadom k tomu, že sme na projekt zohnali nejakých externistov (ktorí nám robili historický výskum a vytvárali grafický content — skrátka neprogramátorskú prácu), prišlo celkom vhod mať fóra, kde sa dá diskutovať o veciach, ktoré nepatria do bugtrackingového systému a riešiť ich cez mail by bolo príliš ťažkopádne. Fóra sme s postupom času prestali používať (hlavne preto, že počet externistov radikálne klesol), ale na začiatku prišli vhod.

- Je celkom pekné spraviť k projektu webstránku, aspoň nejakú jednoduchú. Ak už nič iné, tak máte niečo, čím môžete navonok prezentovať svoj projekt.

- Zamyslite sa aj nad nejakým jednoduchým mailing listom — dobrá cesta, ako doručiť novinky o projekte všetkým zainteresovaným. Opäť je to možno trochu také naše špecifikum, kvôli externistom (vrámci tímu si môžete čokoľvek povedať na schôdzach).

- Ak píšete v C++, tak sa bude tiež hodiť Doxygen (systém na automatické generovanie API dokumentácie zo zdrojákov, ekvivalent JavaDoc-u). Donúti vás to aspoň trochu komentovať svoj vlastný kód (čo po polroku práce oceníte), a jeho použitie je jednoduché (stačí na to napísať jednoduchý skript sa môže sa to robiť automaticky každý týždeň).

- Už som zmienil, že ak robíte na viacerých platformách, bude sa hodiť Cmake. Neviem posúdiť náročnosť jeho administrácie, ale ako užívateľ si naň po chvíli zvyknete.

- Na písanie programátorskej a užívateľskej dokumentácie sme použili LaTeX— tí, ktorí ho poznajú asi nebudú váhať, a vy ostatní — nebojte sa toho, za chvíľu sa do toho dostanete. Pre Windows je dobrá kombinácia MiKTeX (distribúcia) a TeXnicCenter (editor). Bude sa hodiť aj niečo na kreslenie vektorovej grafiky — doporučujem Ipe (lepší na schémy a diagramy) alebo Inkscape (vhodnejší na grafiku).

- Last but not least — hosting. Tu je to jednoduché — potrebujete ho čo najspoľahlivejší. Počas práce na projekte má v zásade tendenciu zlyhávať prakticky všetko, čo aspoň trochu teoreticky zlyhať môže. Spomínam to z dôvodu, ktorý sa pravdepodobne bude medzi informatikmi z MFF tradovať ešte celé generácie — áno, je ním pád urtaxu (server na ktorom KSI okrem iného hostuje všetky projekty) v apríli 2010. S týmto pádom odišlo všetko, čo na projekte bolo zdieľané — SVN, Wiki, projektové stránky a možno aj niečo iné. Väčšinu vecí sa nám samozrejme podarilo zrekonštruovať z lokálnych kópií (čo nám zabralo vyše 2 týždne práce, z toho len týždeň sme čakali, kým administrátor serveru prestane mystifikovať a vylezie s pravdou o probléme von; ďalší skoro týždeň nám zabralo, kým sme odstránili chyby, ktoré vznikli z komplikovaného mergu lokálnych kópií do SVN-ka, pretože sa v nich za 2 týždne samozrejme nahromadilo dosť práce). Napriek tomu sme prišli o celú Wiki (ktorá sa edituje priamo na webe, takže nemá nejakú priamu cestu, ako ju zálohovať mimo ručné ukladanie HTML stránok či ich kódu), celé štatistiky SVN a ešte nejaké ďalšie dáta. Tým nechcem zavrhnúť urtax, len naznačiť jeden fakt — akýkoľvek hosting máte, vaše dáta nie sú nikdy v 100% bezpečí, takže pravidelne vytvárajte aj lokálne kópie všetkého, čo ešte budete potrebovať.

Možno to nie je úplný výčet, ale momentálne ma nič ďalšie nenapadá. Aj keď to môže vyzerať ako spústa práce, nie je to až také zlé, pretože väčšina z týchto technológií je po úvodnom nakonfigurovaní prakticky bezúdržbová. Ešte je dobré dodať, že všetky spomínané technológie a software sú dostupné zadarmo.

**Príprava** Do prípravy vlastne spadá aj výber členov tímu a technológií, ale to by jej popis potom zaberal polovicu tohto dokumentu. Podstatné je, že by ste sa ešte stále mali pohybovať v čase pred nahlásením projektu — do tejto chvíle vlastne vaše aktivity mali hlavne spočívať v diskusií o tom, čo padlo doteraz, a teda by ste nemali za sebou mať viac ako pár týždňov práce. Preto je dôležité sa v tejto fáze zamyslieť, či má vôbec zmysel pokračovať, alebo bude lepšie začať odznova (či už preto, že tím sa ukázal ako nie úplne šťastne vybraný, alebo preto, že by sa váš projekt ukázal technologicky príliš náročný). Aj keď by sa vám do toho odznova nechcelo, majte na pamäti, že strata tých pár týždňov je vždy lepšia, ako strata trištvrte roka.

Ak sa zhodnete, že to zmysel skutočne má, tak je načase si aspoň zbežne preštudovať technológie, s ktorými budete pracovať. Je samozrejme zbytočné sa učiť na spamäť referenčné manuály, ale je dobré získať aspoň hrubú predstavu o designe a funkciách knižníc a frameworkov, ktoré hodláte použiť (ak ju teda ešte nemáte).

Taktiež je potrebné dobre si rozmyslieť architektúru vášho diela. Pri jej návrhu majte na pamäti, že ste tím a je nutné váš software navrhnúť tak, aby bol rozdelený na moduly, ktoré následne budú mať

na starosti jednotliví členovia tímu. Nie je to vždy jednoduché, ale je to dôležité, aby ste zabezpečili, že každý člen tímu bude zamestnaný a nikto nebude čakať na dokončenie určitej časti niekým iným. V našom prípade bola táto časť pomerne zjavná, vzhľadom na to, že sme robili herný engine; projekt sme dekomponovali na grafický, fyzikálny, AI a herný (starajúci sa o samotný herný obsah) modul a na jadro, ktoré spolu tieto moduly integroval dohromady. Tým sme zabezpečili, že každý člen tímu mal na starosti primárne jeden takýto modul a mohol na ňom pracovať skoro nezávisle na ostatných. Je nanajvýš dôležité, aby ste architektonický návrh vášho diela (rovnako ako aj všetko ostatné) poriadne zdokumentovali, aby ste sa k nemu mohli vracať a upravovať ho. Na to je ideálna práve Wiki.

Vedúci tímu by taktiež mal spolu s vedúcim projektu vypracovať časový plán. Názov hovorí sám za seba — jedná sa o časové rozvrhnutie práce na nasledujúcich 9 mesiacov. Zostaviť niečo také bez predchádzajúcich skúseností je samozrejme veľmi ťažké (ako sme sami zistili) a je prakticky vylúčené, že sa vám podarí vytvoriť niečo, podľa čoho naozaj pôjdete s chirurgickou presnosťou (už len preto, že sa nedá dopredu povedať, koľko dní v týždni nad projektom jednotliví členovia strávia). Napriek tomu to má zmysel robiť, pretože vás to donúti si v hlavách zrovnať, čo všetko v projekte chcete mať a umožní vám to zabezpečiť, že to celé tvorí konzistentný funkčný celok.

Kto už niekedy pracoval v tíme vie, že je vhodné sa zjednotiť na tom, ako budete zapisovať kód. To sú na prvý pohľad somariny ako indentácia, názvové konvencie identifikátorov premenných, funkcií a štruktúr, umiestňovanie zátvoriek, medzier a komentárov a tak ďalej. Napriek zdanlivej banálnosti vám to pomôže vyznať sa v kóde nielen vašich kolegov, ale aj vašom vlastnom, pretože po nejakých 6 mesiacoch aj na svoj vlastný kód budete pozerať s nedôverou. Opäť platí — konvencie si na nejaké dostupné miesto zapíšte a hlavne ich aj dodržiavajte...

Posledný bod prípravy je administratíva. V prvom rade je nutné vypísať špecifikáciu pre komisiu. Pre vlastné dobro je vhodné nesnažiť sa o čo najbombastickejší výčet featur budúceho SW diela, ale naopak, držať sa pri zemi a byť (aspoň naoko) skromní. Uvedomte si, že pri obhajobe sa komisia bude čiastočne ohliadať práve na špecifikáciu — a tu platí, že funkcionalita naviac vám kredit zvýši, naopak ak nestihnete podstatnú časť toho, čo ste špecifikovali, tak s veľkou pravdepodobnosťou na obhajobe pohoríte (drobné odchýlky ale obhájiteľné sú). Pri zostavovaní špecifikácie vám v slušnej miere pomôže už spomínaný časový plán.

Špecifikáciu po jej spísaní pošlite komisií a dajte jej členom nejaký čas na rozmyslenie. Ak si nie ste istí (rozsahom, formou, čímkoľvek iným), tak sa nebojte obrátiť na niektorého z nich a špecifikáciu s ním konzultovať. Kým komisia projekt schváli, vy máte čas na to, aby ste si prichystali zázemie — zriadili repository, systém na bugtracking, atď. Po schválení projektu komisiou sa k nemu mailom prihlásite a show sa môže začať.

**Vypracovanie** Vypracovanie projektu stojí a padá na dvoch veciach — ako ste sa na jeho riešenie pripravili a akí vytrvalí ste. U toho prvého je to jednoduché — čokoľvek ste zanedbali pri príprave, budete musieť doháňať počas doby určenej na vypracovanie, a to vás pripraví o cenný čas.

Hlavný problém u SW projektu je pomerne zjavný — väčšina matfyzákov ešte nikdy pred ním nepracovala na projekte takého rozsahu (a ak aj áno, tak prevažne v nejakej firme ako junior programátor). Zápočťáky a ročníkový projekt so SW projektom zrovnanie neznesú — človek na nich pracuje sám a trvajú podstatne kratšie. Do rozsahu sa pridávajú ďalšie faktory, ktoré prácu komplikujú. Prvým z nich je to, že aj keď sa jedná o tímový projekt, tak prakticky nemáte priestor, kde by ste na projekte pracovali spoločne. Možno sa to nezdá, ale predstavuje to dosť veľký problém, pretože riešiť niečo cez ICQ je bolesť a ani Skype vám nenahradí osobnú diskusiu s kolegom. Na tento účel by mohol slúžiť lab, bohužiaľ na počítače tam nemáte administrátorské práva, takže akonáhle budete potrebovať nainštalovať niečo, čo tam nainštalované nie je, ste nahraní (ako je to s Linuxovými strojmi, to neviem). Do úvahy neprichádza ani žiadny neštandardný hardware — v našom prípade by sme k vývoji potrebovali solídne grafické karty, a tie v labe sú len na malom percente počítačov. Ak aj máte to šťastie, že všetci členovia tímu majú notebooky, ktoré si do labu môžu doniesť, tak prichádza na radu ďalší problém, a to nájsť spoločný čas, kedy sa dá prísť všetkým. Cez týždeň sa vám to skoro určite nepodarí a cez víkend sa vám na to polovica tímu vykašle.

Ďalšia vec je motivácia. Akokoľvek vás téma projektu priťahuje, skôr či neskôr aj tak zistíte, že sa vám do toho nechce. V zamestnaní ste za prácu platení, tu nie (pokiaľ nespolupracujete s firmou, a ak aj áno, tak je veľká pravdepodobnosť, že vás nevykompenzujú adekvátne). Hlavnou motiváciou by

malo samozrejme byť dokončenie štúdia, ale to je stále pomerne vzdialený a abstraktný cieľ (akokoľvek to znie divne, možno prídete na to, čo tým myslím). Situáciu zhoršujú aj skúškové obdobia (minimálne jedno vás určite zastihne) — za ten mesiac z toho skrátka vypadnú všetci a potom je to ako prebúdzanie sa z bezvedomia.

Pre toto všetko je na dokončenie projektu esenciálna jedna vec — zachovanie kontinuity práce. Akonáhle vypadnete z pracovného nasadenia, bude čím ďalej tým ťažšie sa doňho dostať. Preto vo vlastnom záujme hneď od začiatku trvajte na pravidelných schôdzach tímu, najlepšie každý týždeň. Je možné, že toto po vás bude chcieť už vedúci, a ak nie, tak to zorganizujte sami a chcite po ňom, aby sa ich zúčastňoval. Dbajte na čo najmenej výpadkov — vopred sa dohodnite na čase, kedy sa bude dať všetkým (rátajte tak s dvoma hodinami na schôdzu) a starajte sa o to, aby absencie boli minimálne.

Schôdza pre vás bude predovšetkým určitá náhrada za to, že spolu nepracujete pod jednou strechou. Na schôdzi by ste zakaždým mali robiť 3 veci:

- Prejdenie práce a všetkého, čo mali jednotliví členovia tímu stihnúť od minula. Každý by mal riadne predviesť a vysvetliť, čo spravil, a ak niečo nestihol, tak vysvetliť prečo. Toto je ideálna zbraň na flákačov — každý si dobre rozmyslí, či sa postaví pred kolegov a znova si bude vymýšľať výhovorky, prečo je pozadu. Dajú sa dokonca zaviesť aj vtipné penalty — viem o kolegoch, čo sa dohodli, že kto nespraví, čo mal, a rozumne to neodôvodní, tak donesie čokoládu. Komu by sa už chcelo na každú schôdzu nosiť čokoládu :)

- Rozdelenie práce do ďalšej schôdze. Každému členovi by malo byť jasné, na čom má pracovať — nič v zmysle „Však ja viem, čo mám robiť“ neprichádza do úvahy. Pridelenie práce by malo prebiehať diskusiou vedúceho projektu alebo vedúceho tímu a konkrétneho človeka, ktorého sa to týka. Množstvo by samozrejme malo byť adekvátne dobe do ďalšej schôdze. Pri zadeľovaní práce vám tiež pomôže časový plán. Všetko, čo sa dohodne, zapíšte do systému na bugtracking ako nové tasky (pričom sa snažte o čo najjemnejšiu granularitu a konkrétnosť — „Naprogramuj databázový modul“ nie je task, ale rola).

- Riešenie problémov. Väčšinou sa bude jednať o technické veci, ktoré členovia tímu nevedia vyriešiť. Môže sa jednať o problém s nejakou technológiou, alebo so zakomponovaním nejakého svojho kódu do existujúcej časti projektu. Všetci členovia tímu by mali zo schôdze odísť s tým, že vedia, čo majú robiť a aj ako to budú robiť (aspoň na úrovni toho, že zvyšok si už sami dohľadajú).

Je rozumné robiť zo schôdze zápisky, aby sa vedelo, čo sa prejednalo, a čo má byť hotové do budúcej schôdze. To vám samozrejme pomôže zistiť, či všetci stihli, čo mali. Opäť platí, že zápisky zo schôdze patria na Wiki. Ďalej, schôdzu je dobré robiť v miestnosti s projektorom, ideálne SW1 či SU1. Večer v týchto miestnostiach väčšinou nikto nebýva a tak vám služba miestnosť bez problémov otvorí a dá ovládač od projektoru.

Ďalšia dôležitá vec je životný cyklus projektu. Každý projekt by mal mať aspoň 3 hlavné fázy, a to Alpha, Beta a Release Candidate. Tieto pojmy by každý už mal poznať, takže zbežne: Alpha je bod, kedy by mala byť hotová kostra a základné časti programu, takže je na ňom možné pracovať v plnej miere. Beta by mala byť bodom, po ktorom sa už nebude pridávať žiadna väčšia funkcionalita, ktorá by mohla program výraznejšie rozhodiť, maximálne nejaké menšie feature. Naopak by sa mali začať v plnej miere začať riešiť známe bugy (ktoré sú zaznamenané v bugtrackingovom systéme). Release Candidate znamená, že všetko, čo ste chceli v projekte mať tam už je a takisto väčšina známych bugov je už opravená (používa sa aj termín „feature freeze“).

Dĺžka týchto fáz nie je pevne daná a samozrejme závisí na charaktere vášho projektu (ak napríklad skeletálna časť projektu bude malá a hlavná časť práce bude na jeho funkcionalite, bude Alpha verzia hotová pomerne rýchlo a naopak väčší dôraz bude na Betu). Feature freeze by mal nastať ideálne mesiac pred odovzdaním, najneskôr však 2 týždne pred ním. Naviac majte na pamäti, že vás ešte čaká napísanie dokumentácie, čo vám zhruba 2–3 týždne zaberie.

Čo patrí do dokumentácie by ste už mali vedieť, takže opäť zbežne:

1. Užívateľská — ako váš systém používať. Vždy ju píšte s ohľadom na znalosti cieľového užívateľa.

2. Programátorská — predstavte si, že po vás príde iný tím a projekt po vás prevezme. Toto je text, ktorý je určený pre nich, aby získali pochopenie jeho vnútorností. Mala by tu byť zachytená architektúra, abstraktný popis modulov, formáty dát a aj myšlienkové postupy vás samých pri zostavovaní toho celého.

3. Projektová — odlišná od programátorskej. Patrí sem popis tímu a jeho zodpovedností, časové rozvrhnutie prác, porovnanie s podobnými projektmi, zoznam známych nevyriešených bugov atď. Dalo by sa povedať, že sa jedná tak trochu o pohľad manažmentu.

4. Dokumentácia API — v zásade to, čo vám vygeneruje Doxygen či JavaDoc. Túto nie je potrebné odovzdávať v papierovej podobe.

Dokumentáciu vám rozhodne neradím odfláknuť, pretože komisia ju skutočne číta (aspoň zbežne) a riadi sa ňou. Sú projekty, ktoré neboli obhájené majoritne práve kvôli dokumentácií.

Keď sa bude blížiť dátum odovzdania, je nutné opäť vyriešiť nejakú tú administratívu. Najneskôr 2 týždne pred odovzdaním sa musíte na adresu komisie prihlásiť na obhajoby (radšej sme nezisťovali, čo by sa stalo, keby sme chceli kvôli získaniu času preskočiť náš termín obhajob). Samotné odovzdanie zahŕňa DVD s projektom, zdrojákmi a dokumentáciou a samotnú dokumentáciu v papierovej podobe — a to celé dvakrát. Doporučujem to všetko vložiť do nejakých dosiek, nech sa to pri transporte nerozsype. Po úspešnom odovzdaní máte presne 2 týždne na trochu oddychu a hlavne prípravu finálnej obhajoby.

**Obhajoba** Obhajoba je vec, ktorú bohužiaľ veľa matfyzákov podceňuje, pretože sa už nejedná o hard-skill programátorskú prácu. To je ale veľká chyba! V skutočnosti by som si trúfol odhadnúť váhu obhajoby na nejakých 30–40%. Môže sa to zdať nefér, že tých 20 minút rečnenia plus týždeň prípravy tvorí tak podstatnú časť niečoho, čo ste tvorili bezmála rok, ale už je to raz tak a je nutné sa s tým zmieriť. U komerčných projektov je to zrovna tak — vydavateľ alebo zákazník projekt zamietne, pokiaľ ho nepresvedčíte, že mu to za jeho peniaze stojí. Pričom tu nejde o žiadne klamstvá alebo povrchnosť — ide o to prezentovať vaše technicky zložité dielo ľuďom, ktorí ho vidia po prvýkrát v živote, naviac nemusia nutne mať technické vzdelanie. V dobe, kedy prakticky vždy máte vedľa seba konkurenciu, je nutné potenciálneho investora najskôr zaujať.

U SW projektu tento účel tvorí práve obhajoba. Pritom „obhajoba“ je tak trochu nevhodné slovo, pretože evokuje dojem, že sa proti niečomu budete brániť. Pritom vašou úlohou je však práve to, čo som už naznačil — postaviť sa pred vaše publikum a zrozumiteľne im prezentovať vaše dielo. V tomto prípade budú tým publikom vaši kolegovia obhajujúci pred či po vás, no a samozrejme samotná komisia (ako morálnu podporu si môžete zavolať aj niekoho ďalšieho, obhajoby sú verejné). Ku komisií sa treba stavať s rozumom — nejaké démonizácie nie sú na mieste, tí ľudia tam nie sú preto, aby vás apriórne napádali. Na druhej strane čakajte objektívnu kritiku, ku ktorej sa treba postaviť čelom. Viem o projekte, ktorý nebol na prvýkrát obhájený, pretože členovia komisie z prezentácie nerozpoznali rozsah vykonanej práce a označili ho za prácu na dva víkendy. Pritom práce tam bolo dosť, lenže navonok zle viditeľnej, a chyba obhajujúcich bola, že toto nedokázali správne vysvetliť. Preto sa nebojte komisií protirečiť, ak dôjdu k nejakým chybným záverom.

Samotná obhajoba vyzerá nasledovne: každý tím dostane 30 minút, za ktoré sa musí projekt odprezentovať, musia sa zodpovedať otázky komisie a naviac potrebuje komisia pár minút na poradenie o vyhodnotení. Takže samotná prezentácia by v žiadnom prípade nemala presiahnuť 20 minút. Žiadne „konvencie“ ohľadom prezentácie nie sú definované — jediná požiadavka je, aby tím predviedol projekt na projektore naživo.

Najprv je potreba rozhodnúť sa, kto bude prezentovať. Možností máte viacero — od postupného vystriedania sa celého tímu až po jedného človeka. V našom prípade prezentoval jediný človek, ja, pretože som z celého tímu mal s prezentovaním najviac predchádzajúcich skúseností. Ak teda máte v tíme takéhoto človeka, pošlite ho tam, vyplatí sa to. Pritom celý tím môže stáť vedľa neho a odpovedať na dotazy komisie, ak sa budú týkať niečoho, na čom sám prezentujúci nerobil (u nás sa to zhodou okolností nestalo, všetky otázky smerovali na grafiku, ktorú som mal na starosti).

Najhoršia vec, ktorá sa môže stať pri prezentácií je zlyhanie techniky, či už hardwaru alebo softwaru. Predíďte tomu, ako to len bude možné. Najlepšie deň vopred si požičajte kľúč od miestnosti, kde sa

budú obhajoby konať, a vyskúšajte si projektor (či už použijete erárny počítač, alebo vlastný NB). Otestujte všetko — slidy, samotný program a prípadne ďalšie materiály, napríklad video. Rozmyslite si, čo chcete vo vašom programe komisií ukázať a tieto veci si dopredu vyskúšajte, jednak aby ste predišli nejakému váhaniu a zmätkom („Tak čo by sme im ešte ukázali?“), a potom aby ste minimalizovali pravdepodobnosť pádu vášho diela pri prezentácií.

K samotnej prezentácií sa nejak košato vyjadrovať nebudem. Existuje množstvo textov s doporučeniami, ako má vyzerať dobrá prezentácia, takže ak nikto z vás nemá s prezentovaním skúsenosti, je načase si o tom niečo zistiť. Hovorte sebavedome, zrozumiteľne a nahlas, nie príliš rýchlo ani pomaly, držte sa osnovy v slidoch, ale za žiadnu cenu ich nečítajte, pozerajte sa smerom do publika, nie na monitor alebo do zeme — to sú hlavné veci, na ktoré je treba dbať. V tejto súvislosti doporučujem predmet Návrhové Vzory, ktoré momentálne vyučuje pán Zavoral; okrem samotných návrhových vzorov si tam vyskúšate prezentáciu (každý prezentuje jeden naštudovaný vzor) a dokonca k nej aj dostanete cenný feedback v podobe postrehov od každého jedného poslucháča. Čo sa týka výroby slidov, doporučujem si prejsť túto výbornú meta-prezentáciu od Petra Kmocha: `http://cgg.mff.cuni.cz/~kmoch/downloads/presenting.pdf` (zaoberá sa hlavne prezentovaním vedeckého článku, ale to nie je také podstatné, tie veci majú obecnú platnosť).

Prípravu prezentácie neodkladajte na poslednú chvíľu, urobte to aspoň týždeň vopred. Slidy by mali pripravovať len tí ľudia, ktorí aj budú prezentovať. Dôrazne doporučujem si prezentáciu vopred vyskúšať — zavolajte si pár kamarátov a samozrejme vedúceho a v SW1 alebo SU1 im projekt nanečisto odprezentujte. Skúška prezentácie vám nesmierne pomôže pri vychytaní jej chýb, ujasnení si jej časového rozvrhu a nadobudnutí sebavedomia, a pri obhajobe tento rozdiel bude poznať. Komisia to ocení ;)

**Záver** Ak ste sa dočítali až sem — gratulujem. Pôvodne som neplánoval dokument takéhoto rozsahu, ale mojim cieľom bolo hlavne detailne a zrozumiteľne rozobrať celý projekt od začiatku do konca. Keď som sa na projekt pripravoval ja, musel som si prečítať viacero postrehov od rôznych tímov a dávať si dohromady útržky informácií. Preto som chcel spísať postrehy natoľko detailné a obecné, aby ich čitateľ už nebol nútený hľadať doplňujúce informácie niekde inde (zvlášť teraz, keď väčšina postrehov z predchádzajúcich rokov zrejme vzala za svoje s pádom urtaxu). Dúfam, že sa mi to aspoň do určitej miery podarilo. Takisto dúfam, že vás neodradil môj materinský jazyk. Zvažoval som použiť angličtinu, ale myslím, že by to odradilo ešte viac ľudí, ako teraz.

V prípade záujmu navštívte našu stránku `http://urtax.ms.mff.cuni.cz/flyingsamurai`. Nájdete tam samotný runtime projektu, ale časom by tam mali pribudnúť aj ďalšie zdroje, ako jeho dokumentácia, špecifikácia a iné veci, ktoré by vás mohli zaujímať. Aj keď majú k dokonalosti ďaleko, Flying Samurai bol obhájený s bonusovými bodmi, takže môžete predpokladať, že komisia k žiadnej časti našej práce nemala vážnejšie výhrady. Ak na stránkach nenájdete, čo vás zaujíma, alebo máte nejaké ďalšie dotazy, neváhajte mi napísať na adresu uvedenú na titulnej strane.

Veľa šťastia s projektom :)