Time Lapsus – technická specifikace

V tomto dokumentu představíme náš plugin a funkcionality, které pro něj plánujeme.

Cílem projektu je implementace pluginu do prostředí Unreal Engine. Tento plugin by měl výrazným způsobem usnadnit vývoj počítačových her adventurního žánru, a to tak, aby byli vývojáři schopni implementovat co největší část hry přímo v editoru, bez pomoci programátorů. Náš plugin uživatelsky co nejpřívětivější formou umožní návrh struktury průchodu hrou, práci s herními předměty a návrh dialogů s nehráčskými postavami. Budeme usilovat o integraci s Unreal Engine 4 editorem a jeho workflow. Funkcionalita pluginu bude demonstrována vytvořením jednoduché ukázkové adventury.

Zadání projektu

Unreal Engine je nástroj pro tvorbu her. Výrazně urychluje a usnadňuje vývoj, ale neposkytuje funkce specifické různým žánrům her, ty musí programátoři tvořit sami. Hlavním cílem tohoto projektu je vytvořit plugin do programu Unreal Engine, který by urychlil a usnadnil vývoj her žánru adventura. Náš plugin by měl usnadňovat:

1. Návrh a implementaci hlavní struktury adventury. Konkrétně vývojářům nabídne:
    a. Vizuální nástroj pro tvorbu diagramu průchodu hrou (např. podobné state chartům z UML).
    b. Jednoduchou správu a organizaci herních stavových proměnných.
    c. Možnost jasně definovat a vyhodnocovat podmínky pro postup hrou.
    d. Podporu pro ukládání a načítání stavu herních proměnných.
    e. Prostor pro definici herních událostí a možnost je vyvolávat.
2. Správu herních předmětů. Konkrétně bude podporovat:
    a. Jednotný interface pro globální definice předmětů.
    b. Nastavení názvu a ikony jednotlivých předmětů.
    c. Nastavení dalších parametrů předmětu, jako například popis či tagy předmětu (v zadání to byly kategorie, došlo k přejmenování).
    d. Vyvolávání definovaných událostí při sebrání či zahození předmětu
    e. Definování akcí, které se mají stát po zkombinování dvou předmětů.
    f. Správu inventáře, tj. rozhraní pro zjišťování toho, které předměty jsou v inventáři a pro přidávání či odebírání předmětů z inventáře.

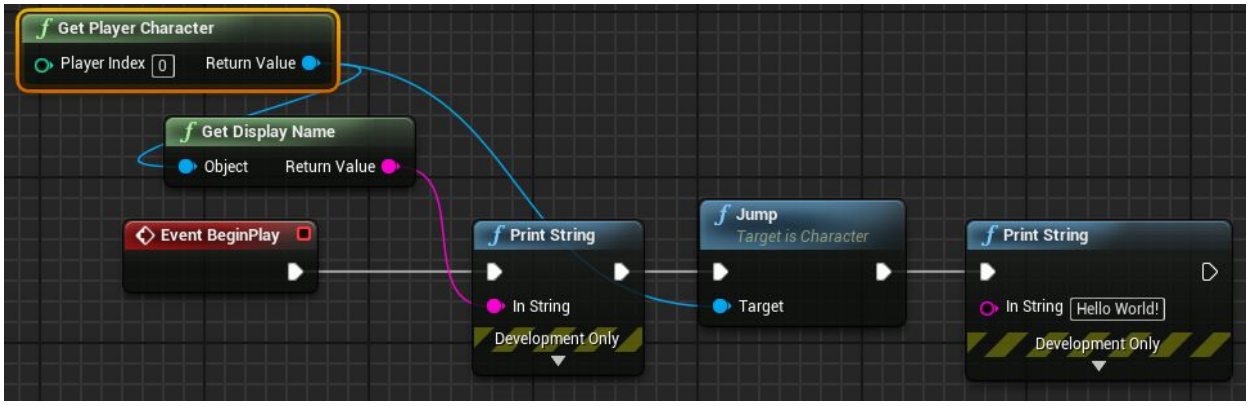
    g. Zobrazení inventáře
3. Definice dialogů. Konkrétně bude podporovat:
    a. Vizuální nástroj pro tvorbu dialogů podobný blueprintům, který by měl sloužit scénaristovi k navrhování dialogů, ve kterém bude možné:
        i. Definovat, které postavy v daném dialogu mluví.
        ii. Definovat repliky postav.
        iii. Vytvořit diagram, ve kterém půjde definovat návaznost jednotlivých replik
        iv. V diagramu udělat podporu větvení podle zvolené odpovědi hráče či na základě hodnot herních proměnných.
        v. V diagramu přidat možnost vyvolání událostí ze Story Engine, viz bod 1.
        vi. Snadno pracovat s proměnnými ze Story Engine, viz 1a.
    b. Zobrazení definovaného dialogu hráči během hry.

Zároveň vytvoříme krátkou adventuru, na které demonstrujeme možnosti našeho pluginu. Nemusí to ale být kompletní adventura, kterou bychom mohli rovnou prodávat, bude nám stačit, když jasně demonstruje všechny slíbené funkcionality.

Celé zadání je k dispozici zde:
https://docs.google.com/document/d/1mCy4AtY764bRMGBaIvzMubZ_8hOQgdgLgzlRzgMcM08/edit?usp=sharing

Unreal Engine

Unreal Engine (zkráceně UE, v aktuální verzi 4) je jeden z nejpopulárnějších herních enginů. Engine je psán v jazyce C++, vývojáři v něm mohou vyvíjet hry jak v jazyce C++, tak za použití vizuálního skriptovacího systému, tzv. Blueprintů. Blueprinty umožňují vývoj počítačových her bez znalostí programování - funkcionalita se skriptuje vizuálně.



Takto vypadá velmi jednoduchý herní skript, který po vyvolání události BeginPlay vypíše do konzole název třídy herní postavy, pak zavolá metodu Jump na instanci hráčem ovládané postavy a pak vypíše do konzole text “Hello World!”. Skripty mohou být i mnohem složitější a lze pomocí nich vytvořit celou hru bez nutností napsat jediný řádek kódu v C++.

Mezi další výhody UE patří mj. veřejně dostupné zdrojové kódy enginu. Také podporuje například zobrazování 2D/3D prostředí, přehrávání zvuků, lokalizaci, multiplatformnost a také nabízí komplexní editor pro vývoj her. Funkce, které v UE nejsou, lze často dodat pomocí pluginů třetích stran.

Definice adventury

Adventura je typ počítačové hry, ve které hráč ovládá svého hrdinu, se kterým se pohybuje herním světem. Tento svět je rozdělen do různých obrazovek, mezi kterými lze přecházet. V herním světě se nalézají různé předměty a další postavy, se kterými může hráč interagovat. Hráč se snaží všechny tyto předměty využít k tomu, aby splnil nějaký cíl hry. Běžnými akcemi, které může hráč během adventury dělat, aby řešil problémy, jsou:

1. Sebrání předmětu – V herním světě se obvykle nalézá mnoho předmětů, které mohou hráči pomoci. Když hráč takový předmět sebere, přidá se mu do inventáře a postava s ním pak může nakládat na jiných obrazovkách.
2. Používání předmětů – Hráč může použít nějaký předmět na scéně či v inventáři. Co znamená použití je specifické pro každý předmět. Například použití knihy může znamenat, že postava knihu přečte, použití spínače znamená, že se stiskne, použití jídla může znamenat, že ho postava sní. Jiné předměty zase nelze samostatně použít vůbec.

3. Kombinování předmětů - Předměty spolu lze také kombinovat, vždy nějaký předmět z inventáře s nějakým předmětem na scéně či s předmětem v inventáři. Například kombinace nože a lana lano přeřízne, kombinace dvou částí mapy vytvoří celou mapu, kombinace vybitého mobilu a powerbanky vytvoří nabitý telefon.
4. Dialog s postavami – V herním světě se obvykle vyskytují různé postavy, které nejsou ovládány hráčem (NPC). S nimi je možné komunikovat. Většinou je s nimi možné začít dialog, ve kterém si hráč může vybírat, jak jeho postava odpovídá na repliky druhé postavy.
5. Prozkoumávání předmětů – Předměty, jak ty na scéně tak ty v inventáři, je v adventurách možné prozkoumat. Což většinou znamená pouze to, že postava o předmětu něco řekne, ať už kvůli humoru nebo kvůli tomu, aby předmět přiblížila hráči. Tato akce většinou není nutná pro postup hrou, ale v určitých případech může být. Například prozkoumání nalezené sekery může prozradit, že původní vlastník si na ni vyryl jméno a že mu ji lze vrátit.

Hráčovým úkolem v těchto hrách je vymyslet, v jakém pořadí má tyto akce udělat, aby postoupil v příběhu dále.

Ve většině dnešních adventur je běžné, že jsou rozděleny do několika kapitol. Kapitoly jsou většinou od sebe odděleny jak příběhově, tak i návrhově. Tj. není možné, aby hráč v 1. kapitole zapomněl sebrat nějaký předmět a musel se pro něj v rámci 2. kapitoly vracet. Většinou mají kapitoly zcela oddělené předměty, co se v nich používají, a problémy, co je v nich potřeba řešit. Stejně tak postavy mají často v různých kapitolách různé dialogy.

Nefunkční požadavky

Většina dokumentu se týká funkčních požadavků tohoto projektu. Nicméně máme tu i tři obecné nefunkční požadavky, které zde explicitně zmíníme.

1. **Důraz na rozšiřitelnost** – Předpokládáme, že vývojáři, užívající plugin k tvorbě vlastních her, mohou mít nějaké specifické potřeby. Proto by pro ně mělo být co nejjednodušší náš plugin rozšiřovat. Počítáme s tím, že rozšiřování pluginu už budou dělat programátoři, nicméně i pro ně by mělo být co nejjednodušší funkcionality tohoto pluginu modifikovat.

2. **Uživatelská přívětivost** – Očekává se, že náš plugin budou používat i neprogramátoři. Všechny funkcionality tohoto pluginu by pro ně měly být co nejvíce přístupné a pochopitelné.
3. **Integrace do Unreal Engine** – Tento požadavek se váže k uživatelské přívětivosti. Očekáváme, že vývojáři, kteří budou s naším pluginem pracovat, už budou mít alespoň základní zkušenosti s tvorbou her v Unreal Engine. Z toho vyplývá, že čím víc budou naše editory dodržovat konvence a vizuální styl Unreal Editoru, tím jednodušeji se v nich bude vývojář orientovat.

Vývoj adventury a vymezení projektu

V této kapitole uvedeme, jaké kroky jsou potřeba k tomu, aby vznikla adventura. A zdůrazníme, co všechno bude náš plugin ulehčovat a co naopak ulehčovat nebude. Předvedeme si to na příkladu jednoho levelu:

1. Designer vytvoří návrh průběhu kapitoly, tj. návrh toho, co všechno musí hráč udělat pro to, aby úspěšně dokončil kapitolu. Díky našemu pluginu může designer tento průběh navrhovat přímo v editoru.
2. Designer s pomocí našeho pluginu vytvoří definice všech herních předmětů.
3. Grafik a designer společně připraví vizuální stránku kapitoly - rozdělí ji do jednotlivých obrazovek, obrazovky nakreslí, propojí mezi sebou, aby mezi nimi bylo možné přecházet a přidá do nich všechny interaktivní objekty a postavy, i když ještě nejsou funkční. Grafika nemusí být finální. S tímto náš plugin nic do činění nemá, zde se budou využívat čistě funkce Unreal Engine v kombinaci s externími grafickými editory.
4. Animátor vytvoří animace, které jsou nutné pro tuto kapitolu. Animace budou využívat nativní animační systém Unreal Enginu, náš plugin do toho procesu nebude nijak zasahovat.
5. Scénárista navrhne první verzi všech potřebných dialogů tak, aby se v nich říkalo vše, co je pro postup hrou nutné. Návrh dialogů bude náš plugin také usnadňovat.
6. Designer a programátor společně vytvoří skripty, které budou řídit samotný průchod hrou - co se má stát v reakci na různé uživatelovy akce. Bude je tvořit hlavně designer, programátor by měl řešit jen věci na designera příliš složité. V tomto náš plugin pomůže jen částečně - skripty se budou stále tvořit v Unreal Engine pomocí blueprintů či naprogramováním v C++, ale některé často používané úkony náš plugin usnadní, jako například zobrazení dialogů, sbírání předmětů nebo definování akcí při používání, prohlížení nebo kombinování předmětů. Vyvolání těchto akcí bude možné jak z kódu, tak z blueprintů.

7. Playtesting. Testeři si kapitolu vyzkouší a vrátí zpětnou vazbu, která se zapracuje. Tento bod pokračuje tak dlouho, dokud testeři nachází dostatečně závažné problémy. S tímto náš plugin nebude mít co do činění.
8. Finalizace grafiky, animací, dialogů, dabingu a obecně všeho contentu. Testování. Zde bude náš plugin užitečný tam, kde byl užitečný i dříve.

Ve zkratce lze říci, že náš plugin bude výrazně usnadňovat návrh průběhu kapitoly, návrh dialogů a definování použitelných předmětů. Tvorbu skriptů také usnadní, ale ne do takové míry - usnadní práci s proměnnými, s událostmi a zjednoduší dodržování navrhnutého průběhu kapitolou. U všeho ostatního se budou využívat existující funkce Unreal Engine, popřípadě vlastní implementace řešení.

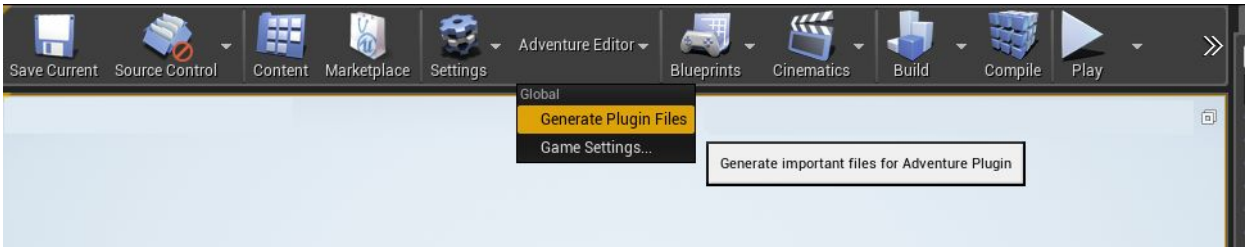
Také je zde vidět, že na projektu pracuje mnoho různých osob s různými schopnostmi. Pro potřeby našeho dokumentu zavedeme pojmy *vývojář* a *programátor*, kde vývojář je kdokoliv, kdo na hře pracuje a nemusí být nutně programátor.

Instalace pluginu

Plugin bude třeba nainstalovat stejně jako jakékoliv jiné pluginy do Unreal Engine. Pluginy se dělí na *editorové* a *projektové*, náš spadá do první kategorie (chceme, aby byl plugin přístupný napříč různými projekty), takže instalace proběhne do složky *Plugins* v instalační složce UE.

Neboť náš plugin pro používání vyžaduje výchozí konfiguraci (například aby hra používala naši GameInstance), do projektu ho bude možné integrovat dvěma způsoby:

- Vytvořením nového projektu námi vytvořeného typu, který zajistí veškerou potřebnou konfiguraci. Součástí řešení tedy bude i vzorová šablona UE projektu.
- Integrací do již existujícího projektu kliknutím na tlačítko Generate Adventure Plugin Files, které po kliknutí vygeneruje všechny potřebné soubory a provede výchozí konfiguraci projektu. Viz mockup.



Po instalaci se zpřístupní veškeré funkce pluginu, tedy:

- Přidá se nový typ Data Assetu (což je obecný soubor s daty) pro ukládání questů a dialogů. Zároveň pro ně zpřístupní námi vytvořené editory.
- Projekt bude moci začít používat všechny námi vytvořené třídy.
- V nastavení projektu přibude sekce s nastavením našeho pluginu.
- Na Game Instance se nastaví defaultní třídy pro práci s inventářem a s dialogy

Všechny tyto funkce jsou podrobně popsány níže.

Adventure Game Settings

V nastavení projektu bude možné provést globální konfiguraci pro celou adventuru, především pak:

1. Main Quest – Který z questů je hlavní. Questy popíšeme níže, jedná se o soubor věcí, co musí hráč udělat pro dokončení určitého úseku hry. V této proměnné je uložen quest, který reprezentuje celou hru.
2. Item tags – Seznam všech tagů, které lze přidělit předmětům. Tagy mohou být například zbraň nebo cennost. Tyto tagy pak lze předmětům přiřazovat. Více je popíšeme níže.
3. Inventory presenter – Výchozí typ třídy pro zobrazování předmětů.
4. Dialogue presenter – Výchozí typ třídy pro zobrazování dialogů.

Game Instance

Skripty na jednotlivých objektech potřebují mít snadný přístup k funkcím, které bude poskytovat náš plugin. Pro práci s objekty, které mají být přístupné po celou dobu běhu hry, se běžně používá singleton `GameInstance`. My vytvoříme jejího potomka `AdventureGameInstance`, která bude navíc obsahovat níže definované třídy, které budou nabízet funkce našeho pluginu herním skriptům:

- Inventory Manager – Třída pro práci s inventářem.
- Quest Manager – Třída pro přístup k jednotlivým questům.
- Dialogue Controller – Třída pro zobrazování konkrétních dialogů.

Toto jsou všechny funkce, které nabízí přímo náš plugin, ostatní se budou řešit pomocí stávajících funkcionalit Unreal Engine. Všechny budou více popsány níže.

Ostatní

Jednotlivé questy a inventář budou umět serializovat a deserializovat svůj runtime stav, což bude nutné pro ukládání a načítání stavu hry. Očekáváme, že příslušné funkce se budou volat při standardním procesu ukládání a načítání, který už podporuje Unreal Engine. Bude podporována pouze textová lokalizace výsledné hry, také pomocí lokalizačního subsystému Unreal Engine.

Story Engine

Story engine je souhrnné označení pro nástroje sloužící pro tvorbu příběhu hry. Základní stavební jednotkou tohoto enginu jsou takzvané questy. Každý quest ve hře bude reprezentován jedním data assetem typu QuestAsset.

Quest

Quest v kontextu našeho pluginu označuje nějaký úkol, pro jehož splnění musí hráč vykonat určité akce. Questy mohou mít subquesty, tedy některé tyto akce lze pro přehlednost vyčlenit a vytvořit z nich další quest.

Questy mohou být závislé pouze na úplném splnění svých subquestů. Tj. není možné, aby hráč pro pokračování v jednom questu musel vykonat akci jiného questu, který není jeho subquestem.

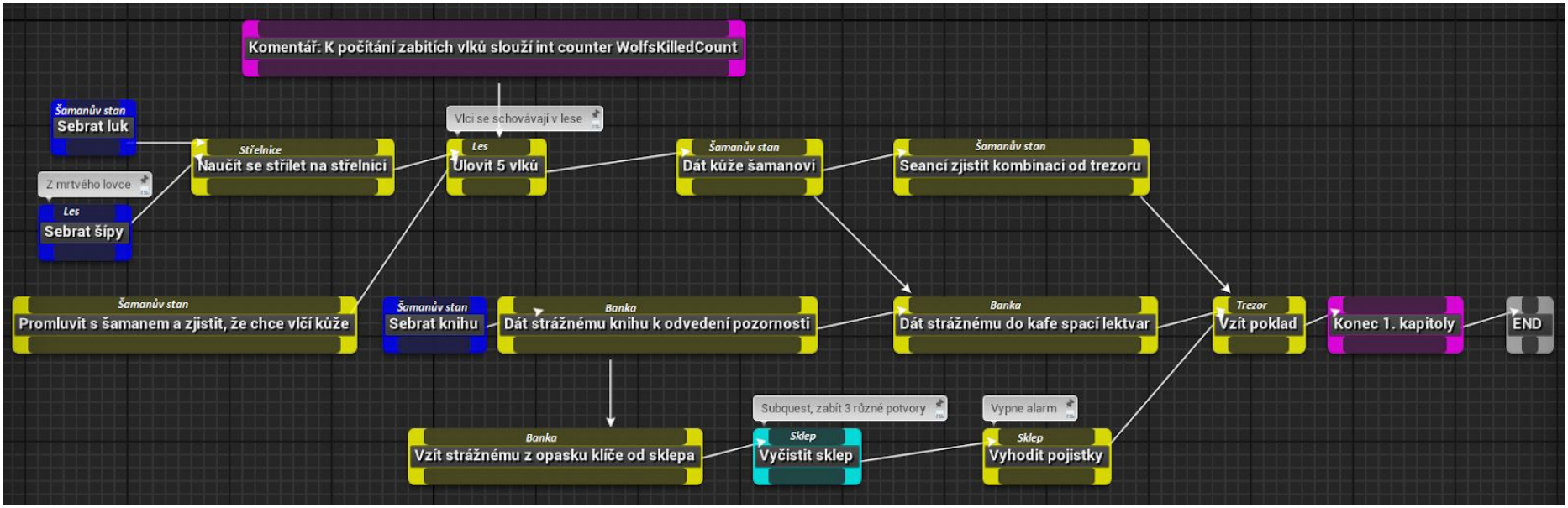
Quest však nemusí však odpovídat tomu, co bychom za quest označili v jiných hrách. Pokud například v rámci kapitoly musí hráč splnit tři různé úkoly, měly by být rozděleny do zvláštních subquestů jen a pouze tehdy, pokud se jejich plnění navzájem neovlivňuje, neboli každý úkol lze plnit bez toho, aby hráč udělal cokoliv v rámci ostatních questů. V klasické adventuře očekáváme, že jeden quest odpovídá jedné kapitole, nicméně není to podmínkou. Očekáváme to proto, že kapitoly jsou na sobě obvykle zcela nezávislé. Ale v rámci kapitoly se často objevují složité závislosti mezi úkoly, co hráč musí splnit.

V questu jsou definovány všechny akce, které hráč musí udělat pro jeho splnění, a proměnné, které jsou k plnění questu potřebné. Story engine však sám nebude tato data měnit – to je zodpovědnost herních skriptů. Jsou to v podstatě jen proměnné s definovanými závislostmi.

Vždy bude ve hře právě jeden main quest, který bude aktivní po celou dobu hry.

Quest Editor

Pro návrh questů vytvoříme grafický editor. Následující mockup znázorňuje, jak si představujeme hlavní editační prostor questu:


Komentář: K počítání zabitích vlků slouží int counter WolfsKilledCount
Šamanův stan
Sebrat luk
Vlci se schovávají v lese
Střelnice
Naučit se střílet na střelnici
Les
Ulovit 5 vlků
Šamanův stan
Dát kůže šamanovi
Šamanův stan
Seancí zjistit kombinaci od trezoru
Z mrtvého lovce
Les
Sebrat šípy
Šamanův stan
Promluvit s šamanem a zjistit, že chce vlčí kůže
Šamanův stan
Sebrat knihu
Banka
Dát strážnému knihu k odvedení pozornosti
Banka
Dát strážnému do kafe spací lektvar
Trezor
Vzít poklad
Konec 1. kapitoly
END
Subquest, zabít 3 různé potvory
Vypne alarm
Banka
Vzít strážnému z opasku klíče od sklepa
Sklep
Vyčistit sklep
Sklep
Vyhodit pojistky

Quest reprezentujeme acyklickým orientovaným grafem závislostí. Každý vrchol představuje nějaký hráčův cíl, který může být buď splněn, nebo nesplněn. Vrcholům budeme říkat Flag. Pokud vede z vrcholu A hrana do vrcholu B, znamená to, že A musí být splněno předtím, než je splněno B. Je obvyklé, že vrchol, který je jednou splněn, už nikdy nemůže změnit na nesplněn, budeme s tím tedy počítat a nedovolíme nastavování Flagů na false. Graf může obsahovat tyto typy vrcholů:

- Simple Flag vrchol – V mockupu je žlutý. Reprezentuje bool proměnnou, která je zpočátku false. Na true ji může nastavit jakýkoliv skript. Na false nastavit nelze.
- Is In Inventory vrchol – V mockupu modrý. Nastaví se na true v případě, že hráč sebere daný předmět. Nekontroluje se dál ovšem, jestli má hráč předmět stále v inventáři. To už si musí skripty hlídat sami.
- Subquest vrchol – V mockupu tyrkysový. Odkazuje se na jiný quest. Tento Flag bude splněn v okamžik, kdy se splní cílový quest.
- Komentářový vrchol – V mockupu fialový. Je true v okamžik, kdy jsou všechny vstupní hrany true.
- END vrchol – V mockupu tmavě šedý. Existuje právě jeden, je v grafu defaultně a nelze ho smazat. Jakmile jsou splněny všechny vrcholy vedoucí do END vrcholu, je splněn i END Flag a s ním se považuje za splněný celý graf.

Pro každý vrchol pak bude možné volitelně nadefinovat lokaci, ve které jej lze splnit, a ta se pak zobrazí ve vrcholu.

Na každém questu bude možné nadefinovat libovolný počet proměnných. Pokud nebude vybrán žádný vrchol, v Properties půjde definovat proměnné stejně jako na blueprint objektech. Budeme podporovat proměnné typu string, bool, int. Půjde též, stejně jako na blueprintech, definovat vlastní události (EventDispatcher).

Při návrhu samotných questů je doporučeno, aby byly všechny questy nutné pro postup ve hře subquesty main questu (nebo subquest subquestu main questu atd.). Pokud to tak designer udělá, jednoduchým projitím grafu hlavního úkolu, jeho subquestů atd. lze snadno zjistit, v jakém stavu hráč je a které všechny akce může udělat pro postup ve hře. To může být zajímavá informace například v okamžik, kdy by se programátoři rozhodli do své hry implementovat systém nápověd. Aby bylo možné hráči napovědět, je nejdříve potřeba zjistit, na čem se hráč zasekl.

Quest editor bude programátorsky rozšiřitelný hlavně o nové, vlastní typy vrcholů. Všechny by však měly stále plnit podmínku, že jakmile jsou jednou splněné, jsou splněné navždy.

Quest Manager

Pro práci s questy během runtime bude na hlavní GameInstance k dispozici třída Quest Manager, poskytující reference na konkrétní questy ve hře.

Na questu bude možné získávat a pokud možno i měnit hodnoty jednotlivých proměnných a vrcholů. Zároveň bude možné poslouchat události na questu.

Quest Manager pak bude poskytovat i seznam všech splnitelných Flagů, tj. těch, jejichž vstupní hrany jsou splněné, ale které samy splněné nejsou. Také bude pro každý vrchol možné získat pravdivostní hodnotu všech vstupních hran.

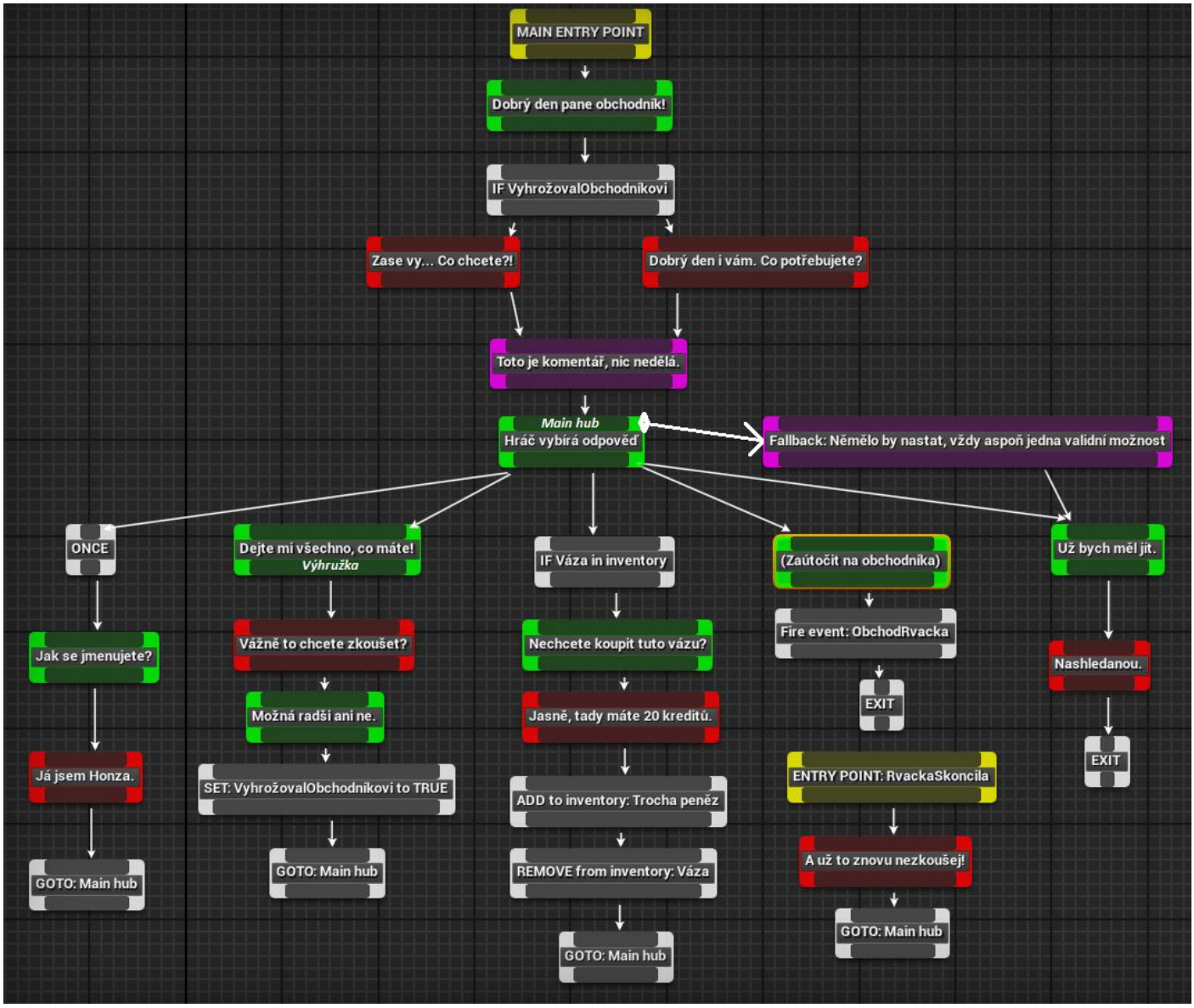
Je důležité zdůraznit, že Quest Manager nebude vynucovat závislosti v grafu vyznačené. Nastavení Flagu na true uspěje i v případě, že závislosti splněny nejsou – taková akce pouze vyvolá warning.

Dialogy

Náš toolset bude podporovat tvorbu dialogů (či monologů). Pro práci s nimi vytvoříme vlastní typ assetů jménem DialogueGraph. Assetů tohoto typu může být v jednom projektu více, v každém z nich pak může být uloženo libovolně mnoho dialogů mezi hráčem a maximálně jednou postavou. Tvorbu dialogů s více NPC najednou náš editor zatím podporovat nebude. Dále implementujeme Dialogue Editor pro tvorbu těchto assetů a Dialogue Controller pro jejich zobrazování během hry. Tuto objekty budou popsány v následujících podsekcích.

Dialogue Editor

Dialogue editor bude podporovat grafický návrh dialogů v assetech typu DialogueGraph. V následujícím mockupu lze vidět příklad hotového grafu pro jednoduchý dialog. Zelené vrcholy reprezentují repliky hlavní postavy, červené vrcholy reprezentují repliky NPC. Fialový vrchol je komentář a šedé vrcholy ovládají průběh dialogu.



Upozorňujeme, že se jedná pouze o mockup, finální editor může vypadat zcela odlišně. Pokud není řečeno jinak, vrcholy mohou mít více vstupních a nejvýše jednu výstupní hranu. Graf může obsahovat tyto typy vrcholů:

- Main entry point – Může být v grafu pouze jeden, nelze smazat, je v grafu standardně po vytvoření nového grafu. Nemůže mít vstupní hrany a má právě jednu výstupní. Reprezentuje místo, ve kterém dialog začne, pokud není explicitně spuštěn z jiného entry pointu.
- Entry point – Jako main entry point, ale není v grafu defaultně, může jich být víc a musí být pojmenován. Reprezentují další místa, ve kterých může dialog začít.

- Player line – reprezentuje repliku hlavní postavy. Je barevně odlišen od ostatních komponent. Musí obsahovat lokalizovatelný text. Následující vlastnosti pak mohou, ale nemusí být definovány:
    - Sound – Reference na namluvený text postavy.
    - Duration – Float v sekundách. Pokud není definován sound, Duration definuje, jak dlouho zůstane text na obrazovce, než sám zmizí. Hodnota 0 znamená, že text sám nezmizí nikdy.
    - Option text – Lokalizovaný string: Když si hráč vybírá, co jeho postava řekne, defaultně se jako možnost zobrazí hlavní text tohoto vrcholu. Pokud je OptionText definován, zobrazí se místo něj.
    - Skippable – Bool, default true. Pokud je true, uživatel má možnost text přeskočit, jinak musí počkat, než doběhne.
- Player line option – Vizuálně identický jako Player line, ale má pouze lokalizovaný text, žádné jiné vlastnosti. Má se použít v případě, že výběr této možnosti nemá vyvolat žádný dialog, ale akci (například rozpoutání rvačky). V mockupu je to vrchol „(Zaútočit na obchodníka!)“.
- NPC line – všechny vlastnosti stejné jako Player line, barevně odlišen od všech ostatních komponent.
- Player line selection – Dává hráči možnost zvolit, co jeho postava řekne. V mockupu vrchol “Hráč vybírá odpověď”. Může mít více vstupních i více výstupních hran. Vrcholům na výstupních hranách říkáme potomci. Potomci jsou uspořádáni. Potomci reprezentují možnosti, které se nabídnou hráči. Proto všichni potomci musí být buď Player line, Player line option, nebo podmínka (či více podmínek za sebou), které po vyhodnocení směřují opět jen do potomků typu Player line nebo Player line option. Kdyby tam byly jiné typy vrcholů, nebylo by jasné, jestli se mají vyhodnotit při zjišťování možností, co se mají zobrazit hráči, nebo až po vybrání té konkrétní odpovědi. Při vyhodnocování tohoto vrcholu se prozkoumají všichni potomci a vyberou se všichni typu Player line a Player line option. Podmínky se vyhodnotí a po platné větvi se pokračuje dále, dokud se nenarazí na vrchol typu Player line nebo Player line option nebo dokud potomek existuje. Všechny tyto vrcholy pak Dialogue Controller nabídne hráči jako možnosti, co jeho postava může říct. Tento vrchol může mít také definovaný fallback, což je vrchol, který se má vyhodnotit, pokud se nenajde žádný platný potomek. V mockupu vede na fialový vrchol Fallback.
- Set quest flag – Nastaví flag nějakého Story engine questu na true.
- Save state – Uloží hru.
- Goto ID – Při vyhodnocení se hned přejde na vrchol s daným názvem.
- Exit – Explicitně ukončí dialog.

- If property equal/less/greater than variable/constant or else – Budou existovat vrcholy pro každý typ proměnné a pro každou možnost z equal/less/greater than a pro porovnávání s konstantami a jinými proměnnými. Pracuje pouze s proměnnými z nějakého Story engine questu. Má dvě výchozí hrany, jednu označenou jako true a jednu jako false.
- Once – Pomocný wrapper kolem If podmínky, která je true právě dokud není poprvé skutečně vyhodnocena. Jinými slovy, nepočítá se vyhodnocování kvůli tomu, jestli nabídnout odpověď hráči v rámci vrcholu Player line selection, ale až okamžik, když hráč tuto konkrétní možnost vybere.
- Comment – Barevně odlišen od ostatních vrcholů, nemá žádnou funkci, pouze dovysvětluje, co se v tuto chvíli děje. Mělo by to fungovat tak, že komentáře u vrcholů komentují jen ten vrchol, komentáře v Comment vrcholu komentují více věci, například kde se dialog odehrává.
- Fire event – Vyvolá požadovanou událost definovanou na daném Story engine questu. Předpokládá se, že některý blueprint ji poté vyřeší. Například blueprint postavy může v reakci na událost změnit animaci postavy či spustit nějaký efekt. Po vyhodnocení eventu se bude pokračovat v dialogu.

Všechny typy vrcholů navíc mohou obsahovat komentář pro podrobnější popsání toho, o co v textu jde (ve spodní části vrcholu, v příkladu je to text „Výhružka“ u druhé volby zleva) a název (v horní části vrcholu, v příkladu text „Main hub“ u výběru odpovědi).

Pokud je třeba uprostřed dialogu provést složitější akci (v mockupu se například spustí rvačka s obchodníkem), musí být vyvolána odpovídající událost přes Story engine (Fire event), ukončen dialog a handling této události na jejím konci musí dialog zase pustit.

Kromě těchto typů vrcholů je možno i definovat několik vlastností na samotném grafu, které se týkají všech vrcholů v tomto assetu:

- Player character – Hlavní postava, která mluví v tomto dialogu. Pokud je v celém projektu jen jedna, použije se právě tato postava.
- NPC – Postava, se kterou hráč mluví. Pokud není definovaná, předpokládá se, že v tomto souboru budou jen monology hlavní postavy.

Bude kladen důraz na to, aby případné rozšíření Dialogue Editoru o nové typy vrcholů bylo co nejjednodušší.

Dialogue Controller

Dialogue Controller bude součástí runtime, přístupný přes GameInstance. Blueprint skripty pak budou moci spustit dialogy přes Dialogue Controller. Vyberou DialogueGraph, který se má pustit, entry point, ze kterého se má začít, a zbytek už obstará Dialogue Controller.

Dialogue Controller si inicializuje Dialogue Presenter a naváže se na jeho události (např. zvolení odpovědi). Pak zobrazí Dialog Presenter na obrazovku, začne procházet DialogueGraph a bude volat příslušné metody DialoguePresenteru, které text a případný výběr možností nabídne hráči.

Dialogue Presenter

Dialogue Presenter bude mít na starosti zobrazení dialogu hráči. To zajistí 2 hlavní metody:

- ShowDialogueLine - metoda, která zobrazí část dialogu uživateli. Argumenty:
    - Dialogue Line Data: Struktura, co obsahuje vše nutné ke spuštění části dialogu:
        - Text: Text, který se má zobrazit hráči.
        - Audio: Audio soubor, který se má pustit.
        - Duration: Jak dlouho má být text na obrazovce, pokud nebyl zadán audio soubor. 0 pro dialog, co sám od sebe nikdy neskončí.
        - Skippable: Bool, pokud je true, uživatel má mít možnost text přeskočit.
- ShowDialogueOptionSelection - metoda, která dá hráči možnost vybrat další repliku v dialogu. Argumenty:
    - Options: Uspořádané pole stringů definující, z kterých možností a v jakém pořadí má uživatel na výběr.

Jak je naznačeno výše, Dialogue Presenter musí definovat události, které se vyvolají, když uživatel vybere odpověď herní postavy, nebo třeba má přečtený text a chce přeskočit mluvenou část dialogu.

Součástí pluginu bude jednoduchá implementace s možností konfigurace některých proměnných, například obrázku pozadí dialogue boxu, fontu, barvy pro dialogy hráče a NPC, a dalších snadno nastavitelných aspektů. Nicméně stejně předpokládáme, že

vývojáři budou spíše využívat vlastní implementace této třídy a tuto obecnou budou používat spíše jako referenci.

Item management

V adventurách se většinou vyskytuje mnoho předmětů, které hráč může sbírat, prohlížet, používat a kombinovat s jinými předměty. Pro snadnější implementaci této práce s předměty náš plugin nabídne několik tříd.

Adventure Game Item

Každý předmět bude reprezentován jedním blueprintem, dědícím od třídy AdventureGameItem. Na něm budou definovány následující metody a vlastnosti, související s jeho chováním v rámci herních mechanik (nikoliv tedy např. informace o jeho umístění či vzhledu na scéně):

- Name – Lokalizovatelný string, představující název tohoto předmětu.
- ExamineDialogue – Bude obsahovat dvě podproměnné typu DialogueGraph a EntryPoint, definující dialog, který se má spustit při prozkoumání předmětu.
- IsExaminable – Pokud je true, tento předmět lze prozkoumat a má na něm smysl volat akci Examine.
- Examine – Defaultní implementace spustí ExamineDialogue, ale bude možné toto chování překrýt.
- IsUsable – Pokud je true, předmět lze použít sám o sobě, a musí být tedy definována Use akce.
- IsPickable – Pokud je true, tento předmět lze sebrat a vložit do inventáře. Jinak nelze.
- UseActionName – Lokalizovatelný string, představující název Use akce, defaultně nil.
- Use – Use akce tohoto předmětu, defaultně nedělá nic, pokud je IsUsable true, bude třeba vytvořit vlastní implementaci.
- Tags – Tagy předmětů, například zbraň, kniha, mapa atd. Definovány v rámci Adventure Game Settings, zde se vybírají všechny tagy, které předmětu lze přiřadit. Samy o sobě nic nedělají, ale skripty je mohou používat.
- Combinations – seznam všech předmětů, se kterými lze předmět kombinovat. Pokud zde přidáme nějakou kombinaci s nějakým dalším předmětem, zobrazí se tato kombinace i u toho druhého předmětu. Tam ji ale nebude možné

modifikovat, jakákoliv modifikace musí proběhnout na tom předmětu, kde byla kombinace založena. Každá kombinace bude mít tyto argumenty
    - CombinationName – Jméno akce, co se vykoná, defaultně nil. Například nůž a lano může mít název „Přeříznout lano“
    - CombinationActionType – Akce, co tato kombinace představuje. Například nůž + lano může být UseAction, zatímco slovník a mluvící meč může být TalkAction.
    - Combine – akce, co se má provést při této kombinaci. Defaultně nil, je třeba vytvořit vlastní implementaci.
- GetCombinationWithItem – Jako argument bere další předmět, vrátí kombinaci, pokud existuje, nebo nil, pokud neexistuje.
- InventoryIcon – ikona tohoto předmětu v inventáři.
- OnPickup – Event Dispatcher, který se vykoná v okamžik, kdy je tento předmět přidán do inventáře.
- OnRemoved – Event Dispatcher, který se vykoná, když je tento předmět z inventáře odebrán.

Inventory management

Inventář je označení pro všechny předměty, momentálně vlastněné herní postavou. Většinou ho lze nějak zobrazit hráči, lze v něm vidět všechny sebrané předměty, používat je, prohlížet, manipulovat s nimi atd. Práce s ním během hry bude rozdělena na dvě části – Inventory Manager, který bude umět pracovat s tím, které předměty v inventáři jsou, bude umět další předměty přidávat atd., a Inventory Presenter, který bude inventář zobrazovat hráči a dovolí mu s předměty manipulovat. Inventory Presenter bude přístupný skrze Inventory Manager, Inventory Manager bude přístupný přes naší Game Instance.

Inventory Manager

Tato třída bude spravovat, které předměty má v dané chvíli která postava, a podporovat serializaci během ukládání hry. Bude mít proměnné:

- Inventory – Vrátí třídu pro práci s inventářem.
- InventoryPresenter – Vrátí třídu pro zobrazování inventáře hráči.

Samotný inventář bude reprezentován třídou Inventory s následujícími metodami:

- GetItems – Vrátí seznam všech předmětů, které jsou v daném inventáři.
- HasItem – Vrátí true pokud je daný předmět v inventáři.
- AddItem – Přidá daný předmět do inventáře.
- RemoveItem – Odebere daný předmět z inventáře.
- OnInventoryChanged – Event Dispatcher, který se zavolá v případě změny tohoto inventáře.

Inventory Presenter

Tato třída dokáže vizualizovat daný inventář. Bude se jednat o interface s dvěma metodami – ShowInventory (zobrazí daný inventář) a HideInventory (skryje daný inventář). V našem pluginu bude vzorová implementace s následujícími funkcemi:

- Zobrazení ikon všech předmětů v daném inventáři.
- Kombinování předmětů přetáhnutím ikony jednoho předmětu na další předmět.
- Zavolání UseAction po kliknutí, pokud má předmět IsUse na true, jinak zavolání ExamineAction po kliknutí.
- Update inventáře po přidání nebo odebrání předmětu.

Další funkcionality, jako například automatické skrývání inventáře za nějakých podmínek, už jsou specifické pro danou hru a vývojář si je bude muset implementovat sám.

Použití předmětu na scéně

Předměty umístěné na scéně musí umět pracovat s výše definovanými daty z AdventureGameItem. Samotná manipulace s předměty však může být velmi specifická dle návrhu hry. Akce jako prozkoumání či použití předmětu mohou chtít vývojáři vyvolat pravým nebo levým tlačítkem myši, prvním či druhým kliknutím, ještě jinak tomu pak pravděpodobně bude u 3D adventur. Pro dosažení těchto efektů budou tedy moci přímo v Unreal Enginu volat nabízené metody na příslušné instanci AdventureGameItem.

Vzorová implementace v rámci pluginu, určená především pro rychlé prototypování, bude mít pouze následující funkci:

- Po prvním kliknutí na předmět zavolá akci Examine (prozkoumání předmětu). Pokud má předmět isPickable false, bude se tato akce opakovat po každém kliknutí.

- Pokud má předmět isPickable true, po druhém kliknutí předmět přidá do inventáře a odebere ze scény.

Runtime – specifikace

V rámci projektu vznikne krátká adventura, na které demonstrujeme možnosti našeho pluginu. Bude se jednat o klasickou 2D point'n'click (myší ovládanou) adventuru, s jednou hratelnou postavou a na několika obrazovkách. V rámci této ukázky bude předvedena práce s inventářem a předměty, jejich sebrání a použití, konverzace s nehráčskými postavami a plnění jejich úkolů.

Adventuru budeme vyvíjet agilně v průběhu vývoje pluginu, díky čemuž bude možné průběžně v každé iteraci testovat jeho funkcionalitu. Vzhledem k tomu, že se již nejedná o samotný plugin, se však na ni nevztahují slíbené nefunkční požadavky (rozšiřitelnost, uživatelská přívětivost atd.). Také je třeba říct, že mnoho funkcionalit lze využít jen pomocí Manager tříd popsaných v pluginu. Následující funkce náš runtime musí implementovat a nejsou popsány nikde výše, ani je bez větších úprav nepodporuje Unreal Engine:

- Vyvolání dialogu s postavou kliknutím.
- Pohyb po obrazovce.
- Ukládání a načítání hry.
- Spuštění hry v češtině či jiném definovaném jazyce (v našem případě angličtině).

Rozdělení práce

Na projektu je v plánu pracovat agilně, tedy rozdělovat práci do dvoutýdenních sprintů a každé dva týdny dělat sprint review a plánování dalšího sprintu. Očekáváme, že tasky budou vývojářům rozdělovány dle potřeby. Jednotliví programátoři budou mít na starosti různé aspekty vývoje pluginu:

- Pavel Halbich (MFF UK) – Programování runtime aspektů pluginu, hlavně programování Manager tříd, Presenter tříd a dema.
- Matouš Kozma (MFF UK) – Project management.
- Marek Polák (MFF UK) – Programování editoru, hlavně vlastních Asset editorů.
- Petr Janda – Game Design, psaní dialogů a příběh

- Pavel Liška – Grafika, animace.

To ovšem neznamená, že Pavel Halbich a Marek Polák budou každý dělat jen jednu část pluginu a že Matouš Kozma nebude programovat. Marek Polák a Pavel Halbich budou každý za svoji část zodpovědní, měli by mít přehled o tom, v jakém stavu jejich část pluginu je. Očekává se také, že dané části budou více rozumět a v případě neshod o technickém aspektu implementace budou mít poslední slovo. Matouš Kozma bude programovat na té části, která bude mít v té době vyšší prioritu. Ale může se stát, že v případě potřeby budou někdy všichni tři vyvíjet jednu část projektu.

Časové rozvržení

Vytvořit přesný plán projektu je náročné ze dvou důvodů. Zaprvé plánujeme pracovat agilně, za druhé počítáme s tím, že během testování přijdeme na to, že některé funkcionality pluginu nebyly navrženy nejlépe z hlediska uživatelské přívětivosti a že je bude třeba předělat. Nejlepší, co můžeme udělat, je navrhnutí milestonů:

- Konec 2. měsíce – Technická specifikace hotova.
- Konec 4. měsíce – Základní funkcionalita hotova. Plugin nemusí být uživatelsky přívětivý. Musí ale být možné vytvořit quest ve quest editoru, musí jít vytvořit jednoduché větvící se dialogy (zatím pouze pomocí výběru možnosti, ne pomocí podmínek), musí být možné sbírat a používat předměty, ale nemusí fungovat kombinace. Musí být možné číst a zapisovat proměnné do questu.
- Konec 6. měsíce – Funkční požadavky splněny.

Během 7.-8. měsíce pak bude probíhat uživatelské testování, ze kterého jistě vzniknou mnohé požadavky na změny, které budeme implementovat. Runtime bude připravovat ukázkový level.

9. měsíc bude probíhat ladění hry, editoru a příprava závěrečné dokumentace.