Jihočeská univerzita v Českých Budějovicích

Přírodovědecká fakulta

Ústav aplikované informatiky

Metodika herního vývoje

Bakalářská práce

Vypracoval: Karel Formánek

Vedoucí práce: doc. Ing. Zora Říhová, CSc.

České Budějovice

2020

Jihočeská univerzita v Českých Budějovicích
Přírodovědecká fakulta

ZADÁVACÍ PROTOKOL BAKALÁŘSKÉ PRÁCE

Student: Karel Formánek
(jméno, příjmení, tituly)

Obor – zaměření studia: Kriminalisticko-technická činnost

Katedra/ústav PřF JU, kde bude práce vypracována a obhájena: UAI

Školitel: doc. Ing. Zora Říhová, CSc.
(jméno, příjmení, tituly, u externího š. název a adresa pracoviště, telefon, fax, e-mail)

Garant z PřF JU: .....
(jméno, příjmení, tituly, katedra – jen v případě externího školitele)

Školitel – specialista, konzultant: .....
(jméno, příjmení, tituly, u externího š. název a adresa pracoviště, telefon, fax, e-mail)

Téma bakalářské práce: Metodika herního vývoje

Cíle práce:

Cílem práce je zavedení vhodné metodiky vývoje pro herní tým ve firmě Sever Game Studio s.r.o., která se zabývá vývojem herních aplikací, z důvodu zlepšení pracovního postupu na herních projektech kvůli spolupráci odlišných pracovních pozic (grafiků a programátorů).

Práce bude obsahovat analýzu současného postupu herního vývoje, návrh metodiky herního vývoje (pravidla pro spolupráci a komunikaci v týmech, pravidla pro práci programátorů a grafiků, týmové role) a zavedení této metodiky spolu s jejím vyhodnocením. Dále se bude také zabývat volbou vhodných nástrojů na podporu metodiky.

Základní doporučená literatura:

- BUCHALCEVOVÁ, Alena. *Metodiky budování informačních systémů*. Praha: Oeconomica, 2009. ISBN isbn978-80-245-1540-3.
- SVOZILOVÁ, Alena. *Projektový management: systémový přístup k řízení projektů*. 3., aktualizované a rozšířené vydání. Praha: Grada Publishing, 2016. Expert (Grada). ISBN 9788027100750.
- JIRKOVSKÝ, Jan. *Game industry: vývoj počítačových her a kapitoly z herního průmyslu*. Praha: D.A.M.O., 2011. ISBN 978-80-904387-1-2.

Financování práce .....

Školitel práce **doc. Ing. Zora Říhová, CSc.** podpis: .....

U externích vedoucích fakultní garant práce .....podpis: .....

Garant oboru bak. studia (nepožaduje se u oboru biologie) .....podpis: .....

Vedoucí katedry/ústavu PřF JU, kde proběhne obhajoba.....podpis: .....

Případný souhlas vedoucího ústavu AV .....podpis: .....

V Českých Budějovicích dne **27.2.2019** Podpis studenta .....

Bibliografické údaje

Formánek, K., 2020: Metodika herního vývoje [Game development methology, Bc. Thesis, in Czech] – 50 p., Faculty of Science, The University of South Bohemia, České Budějovice, Czech Republic.

Abstrakt

Tato práce se zabývá návrhem metodiky pro vývojářský tým ve společnosti Sever Game Studio, s.r.o. a její zavedení. Teoretická část stručně popisuje náhled do vývoje her a základní principy agilních metodik. Praktická část obsahuje analýzu současného stavu vývoje. Soupis kritérii pro návrh nové metodiky, podle kterých je pak navrhnuta a zavedena metodika spolu s jejím vyhodnocením. Tato metodika zefektivnila spolupráci v týmu a postup vývoje na projektu.

Abstract

The bachelor's thesis deals with methodology design for the development team in Sever Game Studio s.r.o. The theoretical part briefly describes game development and basic principles of agile methodologies. The practical part contains an analysis of the current state of the game development in the company and list of criteria for the design of a new methodology, according to which the methodology is then designed, implemented and evaluated. This methodology streamlined team collaboration and the development process on the project.

Klíčová slova – metodika, agilní metodiky, scrum, herní vývoj, práce v týmu

Keywords – methology, agile methologies, scrum, game development, team work

Prohlašuji, že svoji bakalářskou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury. Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V _____, dne _____

Podpis _____

Rád bych tímto velice poděkoval paní doc. Ing. Zoře Říhové, CSc.. za odborné konzultace a rady v průběhu zpracování této bakalářské práce.

Obsah

1. Úvod

Hry, neboli dříve častěji nazýváno videohry, tu již existují bezmála více jak 50 let. Mnoho firem ve světě se dokonce specializují konkrétně jenom na vývoj her. V současné době s vydáváním nových herních konzolí, se herní vývoj stává čím dál tím více známějším odvětvím na trhu.

Cílem práce je návrh a zavedení nové metodiky pro vývojový tým ve firmě Sever Game Studio s.r.o., která se zabývá vývojem herních aplikací. Důvodem je zlepšení pracovního postupu na herních projektech kvůli spolupráci odlišných pracovních pozic (grafiků a programátorů).

Práce obsahuje analýzu současného postupu herního vývoje, návrh metodiky herního vývoje (pravidla pro spolupráci a komunikaci v týmech, pravidla pro práci programátorů, grafiků, projekt manažérů, zvukařů a týmové role) a zavedení této metodiky spolu s jejím vyhodnocením. Dále se také zabývá volbou vhodných nástrojů na podporu metodiky.

Vývojový tým nebo herní studio se skládá z několika specializovaných členů. Každý člen zastává buď funkci programátora, designéra, grafika, testera, zvukaře nebo producenta. Tyto pozice při vývoji spolu velmi úzce spolupracují, a proto je kritické mít dobrou komunikaci mezi členy a pochopit proces herního vývoje a zároveň je důležité, aby herní studia měly pevnou hierarchii v týmu a plán projektu.

Teoretická část je zaměřena na analýzu herního vývoje, týmových rolí a metodiky plánování projektu.

Praktická část obsahuje analýzu současného vývojového týmu, průběh vývoje v Sever Game Studio s.r.o., návrhy zlepšení, pravidla pro spolupráci týmových pozic a jejich implementaci s vyhodnocením.

2. Teoretická část

Tato část se zajímá složitostí herního vývoje. Jeho etapy vývoje. Důležité pozice v týmu v herním vývoji a nakonec metodiky určené pro tento vývoj se stručně vysvětlenými principy fungování a jejich výhodami a nevýhodami.

2.1 Co je to hra

Hra je interaktivní software, jenž je v dnešní době obrovským fenoménem zasahující do mnoha oblastí. Ať už pozitivně nebo negativně. Každá počítačová hra má svůj vlastní svět, který je ovládán pomocí klávesnice, myši a různých herních ovladačů. Začíná se také používat ovládání přes hlasové příkazy za asistence mikrofonu. Díky ovládacím periferiím je hráč přímo v kontaktu s okolním virtuálním světem a může ho ovlivňovat, jak uzná za vhodné. Musí plnit určité úkoly, které jsou omezené časovým limitem, nebo mají základ v logické hříčce (Trčka, 2010).

2.2 Herní vývoj

Herní vývoj je proces, při kterém se vytváří interaktivní software s herními prvky, které jsou zaměřené na poskytnutí zábavy. Jak bylo zmíněno v úvodu, vývojový tým se skládá z členů specializující se na jinou činnost práce. Ačkoli každý vývojový tým nebo herní studio má svůj harmonogram plánu vývoje hry, z pravidla vždy začíná prvotním konceptem a končí testováním a laděním hotové hry.

Hra se obvykle může vyvíjet několik dní až několik let, záleží na velikosti hry, žánru, propracovanosti, počtu lidí a pro jakou platformu jsou vyvíjené. Například jednoduchá hříčka s názvem Flappy Bird byla vytvořena během tří dnů (Tweedie, 2014), kdežto třetí pokračování z velice známé herní série Diablo 3 trvalo vyvinout 11 let (Baird, 2016)

Nedílnou součástí při vývoji hry je také finanční rozpočet, které si studio určí během plánování. Rozpočet závisí na několika aspektech. Prvním je velikost vývojového týmu. Pro některé herní tituly bohatě postačí jeden člověk a pro jiné, například AAA hry (vysoko rozpočtové hry) může počet přesahovat 1000 vývojářů (Makuch, 2013).

Dalším důležitým aspektem jsou licence. Jak bylo zmíněno dříve, vývoj pro herní konzole jako PlayStation 4 nebo Xbox One zabere více času než vývoj pro mobilní zařízení, většinou z důvodu komplexnosti herních titulů. K vydání hry na některé zařízení (Xbox, PlayStation, Nintendo Switch, iOS) jsou potřeba vývojářské baličky (SDK) a vývojářské účty, které často stojí určitý finanční poplatek (Turkee, 2015)

Mnoho herních studií se řídí podle stejných vývojových fází. První fází je vytvoření konceptu, kde vytvoří několik návrhů a ujasní se představa o hře. Pak následuje předprodukční fáze, v této fázi se začne psát potřebná dokumentace týkající se herního konceptu, takzvaný GDD (Game Design Document), který obsahuje všechny aspekty hry, například příběh, postavy, herní mechaniky, zvuky, atd..) (Chandler, 2013).

V předprodukci se také zhotoví plán postupu vývoje a vytvoří se několik herních prototypů, které slouží k tomu, aby se zjistilo, kterou cestou se má hra ubírat, aby byla co nejzábavnější pro cílovou skupinu hráčů. Jakmile se vývojový tým shodne na finálním konceptu hry, na základě toho, co zhotovili v předprodukci, přesune se projekt do další fáze a to fáze produkční. Tato fáze je nejdůležitější, jelikož všechny aspekty hry, jenž byly dohodnuty a napsány v GDD se začnou realizovat. Jakmile vývojový tým vytvoří veškeré assety, neboli obsah (animace, skripty 3D grafické modely, zvuky, atd.), potřebné pro hru, nastává fáze testování a dolaďování, neboli poprodukční. Přesto, že se hra vyvíjí v těchto fázích, používají se milníky, jimiž se určuje vývojový stav hry. Nejčastějšími milníky jsou (Chandler, 2013):

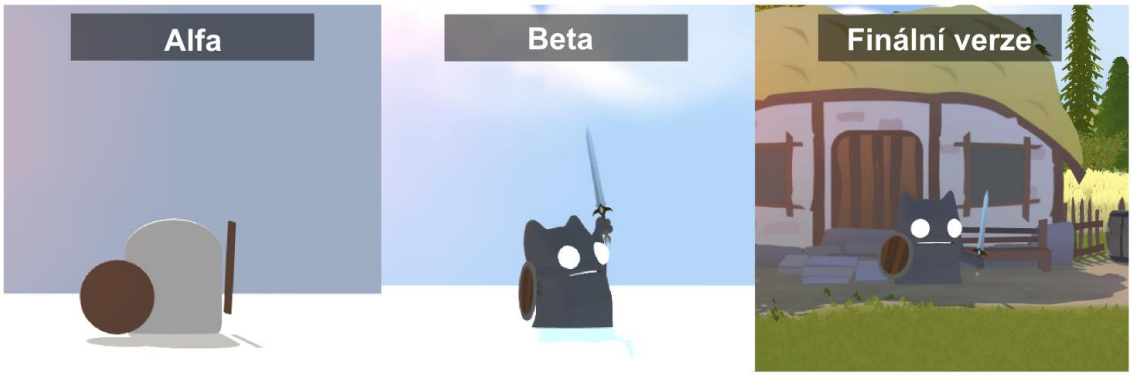
Alfa - verze, kde je jádro hry hotové a hra je již hratelná na herní platformě. Ovšem ne všechny mechaniky a ne všechen herní obsah je implementován.

Beta - verze, kde jsou všechny herní mechaniky již hotovy a opravují se už jen bugy (chyby). V této verzi se už nemění koncept hry.

Open Beta – verze, která je po určitou dobu poskytnuta hráčům, aby ji pro vývojáře testovali. Toto bývá u multi-playerových her, kde se také testují servery, na kterých hra běží.

Closed Beta – stejná jako Open Beta, jen není volně dostupná a je poskytnuta pouze omezenému množství lidí.

Finální verze - verze, v které se hra již může vydat.



Obrázek 1 - Příklad, jak může hra vypadat v určitých verzích vývoje (Zdroj: autor)

2.2.1 Týmové role v herním vývoji

Pokud se podíváme na seznam autorů nějakého herní produktu, uvidíme zde několik pracovních pozic, které se podílely na vývoji. Zde budou vypsány ty nejdůležitější (Shylenok, 2019):

Herní designér - Přináší nápady do hry, vymýšlí herní mechaniky, utváří příběh, navrhuje, jak mají vypadat postavy, prostředí, úrovně, jak se hra bude celkově hrát. Úzce spolupracuje s dalšími pozicemi.

Level designér - Téměř každá hra se skládá z herních úrovní, které hráč prochází. Práce level designéra spočívá v tom, tyto úrovně vytvořit, tak aby byly maximálně pro hráče zábavné.

Koncept artista - Kreslí návrhy postav, prostředí, objektů jako reference pro grafiky, kteří již podle toho vymodelují skutečné 3D modely nebo level designéři, kteří dle tohoto konceptu vytvoří prostředí.

Grafik - Kreslí anebo modeluje 3D objekty, prostředí a postavy pro hru.

Animátor- Přivádí vymodelované postavičky anebo prostředí k životu.

Zvukař - Vytváří zvukové efekty.

Tester - Má na starost testování hry a hledaní chyb.

Projektový manažér - Stará se o chod celého projektu a organizuje jednotlivé úkoly.

Tento koncept týmových rolí se týká velkých společností. Z důvodu zkušeností a počtu lidí ve společnosti Sever Game Studio s.r.o, není tento koncept pro praktickou část uskutečnitelný. Některé role jsou kvůli tomu spojeny.

2.3 Metodiky projektování

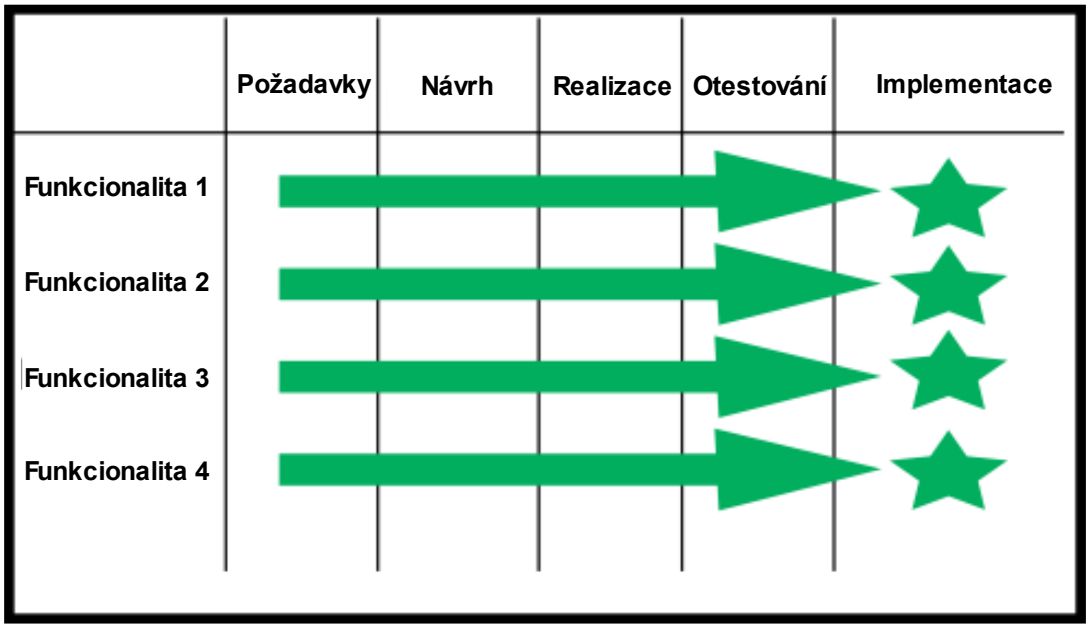
Z důvodu problematiky tvorby hry, vývoji nejvíce prospívá agilní přistup k projektování, jelikož se převážnou část vývoje prototypuje a zkouší se různé herní mechaniky a dopředu se neví, jak přesně bude finální produkt vypadat.

Agilní metodiky pro řízení vývoje software jsou takové metodiky, které pružně reagují na změnu, průběžně rozvrhují práci v průběhu vývoje a ověřují výstupy s uživateli. Agilní metodiky obsahují základní principy, kterými by se měl úspěšný projekt vývoje software řídit. Proces vývoje hry je díky agilnímu přístupu postavený na týmové spolupráci a otevřené komunikaci týmu (Dostupné z: https://managementmania.com/cs/agilni-metodiky-rizeni-vyvoje-software)

Hlavním cílem agilních metodik je minimalizovat risk ve vývoji softwaru díky krátkým časovým úsekům, neboli iteracím, které většinou trvají od jednoho do čtyř týdnů. Každá iterace je jako vývoj menšího projektu, kdy se vyvine malá část aktuálně potřebných funkcionalit (Korkishko, 2017).

Výhody agilní metodiky	**Nevýhody agilní metodiky**
Reagování na změny při vývoji	Možnost, že se bude postrádat práce na dokumentaci
Přímá komunikace týmu a transparentnost průběhu vývoje	Možnost, že změny způsobí větší náročnost
Včasná oprava chyb	

Tabulka 1 - Výhody a nevýhody agilní metodiky (Korkishko, 2017, překlad: autor)



Obrázek 2 - Princip agilních metodik (Korkishko, 2017, překlad: autor)

Scrum

Scrum je agilní metodika, která při správném fungování má vytvořit samo-organizující tým což znamená, že se členové mají aktivně podílet na plánování vývoje a sami si mají hledat řešení na aktuální problémy, v podstatě nemají být pouze pasivními vývojáři, kteří pouze pracují na zrovna přiděleném úkolu. Scrum se definuje jako flexibilní, produktová strategie, kde vývojový tým pracuje jako celek, aby dosáhl společného cíle. Základem Scrumu je rozdělení předem pevně daných funkcionalit v seznamu podle priority. Po každém časovém úseku následuje jejích splnění a otestování. Provádí se každodenní krátké schůzky vývojového týmu (Korkishko, 2017, překlad: autor).

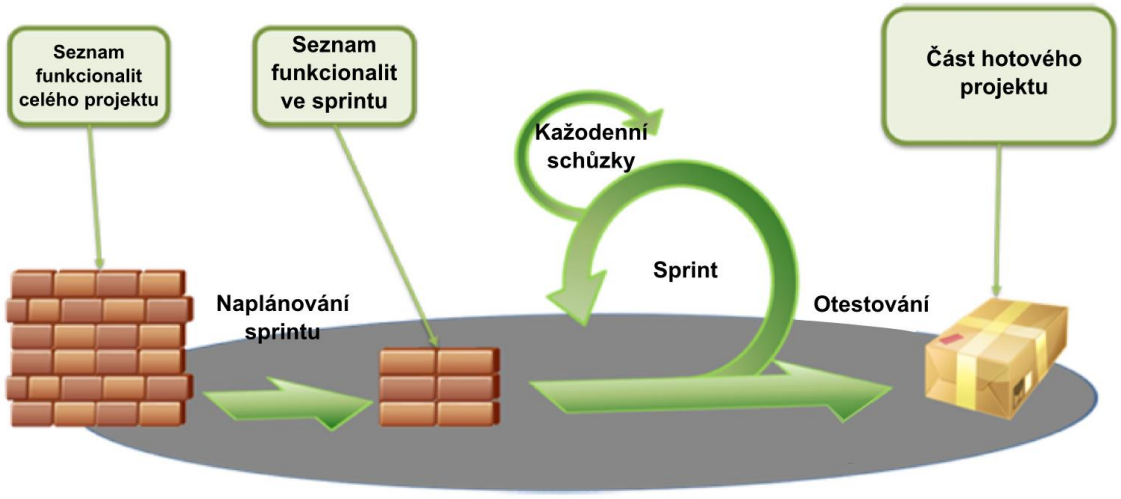
Vývoj pomocí Scrumu zprostředkovává Scrum master, který odstraňuje překážky členům týmu, aby mohli úspěšně dosáhnout svého cíle. Scrum master není lídr týmu, ale spíše se stará o to, aby se týmu dobře pracovalo a má za úkol odstraňovat překážky, které by mohly mít negativní vliv na projekt. (Korkishko, 2017, překlad: autor).

Scrum používá sprinty, což jsou časové úseky (2 až 4 týdny), během kterých se tým zaváže, že se splní určité funkcionality, které se na konci sprintu otestují. Ve sprintu se vždy pracuje pouze na nejvíce prioritních funkcionalitách, které určuje Scrum master na základě potřeb klienta (Korkishko, 2017, překlad: autor).

Výhody Scrumu	Nevýhody Scrumu
Rozhodnutí leží v rukou týmu	Možnost, že se bude postrádat práce na dokumentaci
Dokument o obchodních požadavcích může být považován za zanedbatelný	Není určeno pro velmi velké projekty
Dobře se řídí změny v plánování	Je zapotřebí zkušený tým, Scrum není určen pro nováčky

Tabulka 2 - Výhody a nevýhody metodiky scrum (Korkishko, 2017, překlad: autor)

V současné době se Scrum propojuje i s metodikami jako je PRINCE2 a díky tomu je toto spojení metodik, mimo jiných výhod, vhodné i pro velké organizace čítající přes 400 000 zaměstnanců (Tománek, 2015).



Obrázek 3 - Princip Scrumu (Korkishko, 2017, překlad: autor)

Po konzultacích s jinými herními vývojáři na vývojářských konferencích, konkrétně GameAccess 2018 a 2019, se probíralo, jaké metodiky používají při vývoji na svých projektech. Výsledkem bylo, že žádná speciální metodika se nepoužívá. Firmy vycházejí z agilních metodik, které si pak upravují pro svoje potřeby

3. Praktická část

Tato část se bude zajímat analýzou stávajícího postupu herního vývoje, práce týmu a budou použity postupy z teoretické kapitoly upravené tak, aby prospívaly agilní metodice společnosti pro herní vývoj.

3.1 O společnosti

Sever Game Studio, s.r.o., je společnost, která se zabývá herním vývojem a vývojem aplikací pro virtuální realitu do firemních sektorů. Společnost byla založena v listopadu v roce 2015. O rok později vydala jednoduchou 2D platformovou hru Alpha Wolf na Google play a App store.

Dlouhý vývoj další hry, mířenou zprvu na mobilní zařízení, později i na PC, herní konzole a zároveň i jako desková varianta, se potýkal s dosti vývojovými problémy. Od poloviny roku 2018 se společnost zabývala i vývojem aplikací do virtuální reality a o rok později založila společně s další společností, která se zaměřuje výrazně na virtuální realitu, dceřinou společnost Virtual Lab Development, s.r.o., jenž se specializuje na vývoj VR a AR aplikací, což jsou aplikace pro rozšířenou realitu.

3.2 Proč je nutné zavádět metodiku

Důvod zavádění je prostý, doposud nebyla ve firmě žádná funkční metodika, která by přiváděla nějaké dobře viditelné výsledky. Komunikace v týmu byla často chaotická. Nedělaly se věci, které byly důležité na projektu. Chyběl celkový přehled o stavu vývoje, nebylo plně využito rozložení pracovních kapacit a chyběl celkový plán postupu, tudíž někteří členové týmu pořádně nevěděli, na čem mají pracovat a ani nevěděli, co už je hotové a co zcela chybí. Dále byly někteří členové týmu na vedoucích pozicích zahlceni operativou a tím nemohli dostatečně komunikovat a řídit své členy, které měli na starost.

3.3 Analýza současného postupu vývoje

3.3.1 Z hlediska managementu

V současnosti společnost funguje celkem s 16-ti členy, kteří jsou rozděleni do tří týmů. Hlavní tým se plně zaměřuje na vývoj projektu, druhý tým, který má na starost

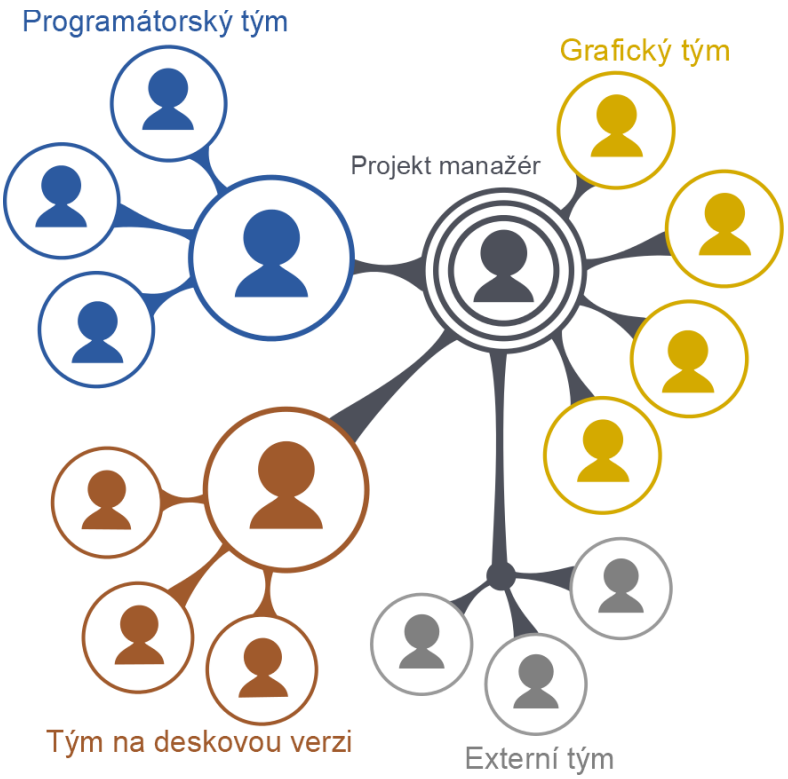
deskovou verzi a pak tým externistů, kteří dostávají úkoly, které nejsou nijak zvlášť kritické, anebo tolik nespěchají na dokončení. Velká většina členů jsou studenti, a proto není možné, aby všichni věnovali převážnou část svého času pouze do vývoje projektu. Zároveň je neslučitelné, aby každý člen zastupoval pouze jednu vývojovou roli z důvodu někdy dostatečných vývojových znalostí členů a z důvodu finančních prostředků.

Vývojovou rolí, v předchozím odstavci, je myšleno, například role programátora, grafika specializujícího se na 2D grafiku či 3D grafiku, zvukaře, animátora, projekt manažéra, designéra herních úrovní, atd. Tudíž v tomto případě je potřeba, aby grafik zvládal jak 2D grafiku, vymodelování 3D modelů a tvorbu jejich textur tak i animací.

Složení týmu

Hlavní tým se skládá ze tří programátorů a čtyř grafiků, z toho dva grafici zároveň dělají i animace. Tým, který vyvíjel deskovou verzi hry měl čtyři členy, z nichž jeden člen se staral zároveň i o PR (Public relations, neboli pozice, kde se dotyčný stará o vztahy s hráčskou veřejností). Poslední tým externistů čítal jak programátory, tak i grafiky.

Pro lepší pochopení sestavení týmu, se hierarchie znázorní v obrázku 4. S ohledem na zachování soukromí členů, je u ikon v obrázku napsána pouze jejich pozice, nikoliv jméno, pro účely této práce nejsou potřebná. Zároveň se tato analýza nebude zaobírat týmem na deskovou hrou, neboť se nejedná svou podstatou o softwarový projekt a také nezasahuje do fungování herního vývojového týmu.



Obrázek 4 - Současná hierarchie (Zdroj: autor)

Ze současné hierarchie je vidět, že projektový manážer má na starost až přílíš lidí, neboli vývojových týmů a přitom sám je členem v jednom vývojovém týmu a tak se musí starat o chod projektu a zároveň funguje sám jako operativec, což znamená, že aktivně vyvíjí a neplní pouze funkci projekt manažéra.

Projekt manažér

Práce projekt manažera v tomto případě spočívala v těchto bodech:

- soupis backlogu, neboli seznamu úkolů projektu do webové aplikace Trello, aby se vědelo, jaké funkcionality v rámci projektu se musí udělat
- udělování úkolu jednotlivým členům
- udělování úkolu sám sobě
- zkontrolování hotových úkolů, zda jsou v pořádku a bez chyb a splňují vše, co bylo třeba v rámci úkolu dodělat a následně přidat, pokud se jedná o grafický nebo zvukový soubor, do projektu, pokud je již daný člen sám nepřídal

- komunikace s celým grafickým týmem, vedoucím programátorem, celým externím týmem a PR členem
- psaní týdenních reportů, což je sepsání sourhnu co se za týden do projektu přidalo, smazalo či změnilo
- uskutečňování schůzek.

Projekt manažer musel komunikovat celkem až s devíti členy a sám si sobě také přidával jednotlivé úkoly. To mnohokrát zapříčinilo, že nestíhal zvládat včas svoji přidělenou práci a přitom přerozdělovat úkoly ostatním, buď z důvodu, že byl zahlcen svojí prací nebo nemohl ihned zkontrolovat hotový úkol a případně se s dotyčným členem dohodnout na úpravách nebo mu přidělít úkol nový. Tak se stávalo, že někteří členové byly například celý týden bez přiděleného úkolu.

Zadávání úkolů

Úkoly jednotlivým členům se přidělovaly při společných schůzích týmu po odsouhlasení dotyčného člena, že tento úkol chápe stejně, jak ho projekt manážer sepsal do popisu úkolu nebo ústně vysvětlil. Často se stávalo, že úkoly byly také přidělovány při soukromé online komunikaci a to z důvodu nepřítomnosti člena týmu na schůzce nebo při zadání nového úkolu po již splněném úkolu. Důležité úkoly projekt manažér zadával členům v grafickém týmu nebo vedoucímu programátorovi, záleželo zda úkol obsahuje grafickou práci nebo programátorskou. Vedoucí programátor si pak tyto úkoly přerozdělil mezi členy ve svém týmu.

Nedůležité úkoly se přidělovaly členům v externím týmu, což byl tým složený z jednoho programátora a dvou grafiků. Důležitost úkolu projekt manažer zakládal na svém předsudku, jak moc si myslel, že tento úkol bude potřeba nyní udělat. Pokud na daném úkolu nestojí jedna z hlavních částí hry, tak tento úkol připadnul někomu z externího týmu. Bohužel, dotazy členů v programátorském týmu byly často směřovány k projekt manažerovi a ne k vedoucímu programátorovi a ten je následně odkazoval na vedoucího programátora, což dospělo později k tomu, že pak tyto úkoly nakonec stejně projekt manažér řešil s dotyčnými sám, místo toho, aby je řešil vedoucí programátor. Tím se, ale obcházela komunikace mezi vedoucím programátorem a členy jeho týmu a

vznikaly problémy, že vedoucí programátor nevědel nejaktuálnější informace ohledně změn detailů v úkolech a práce na nich.

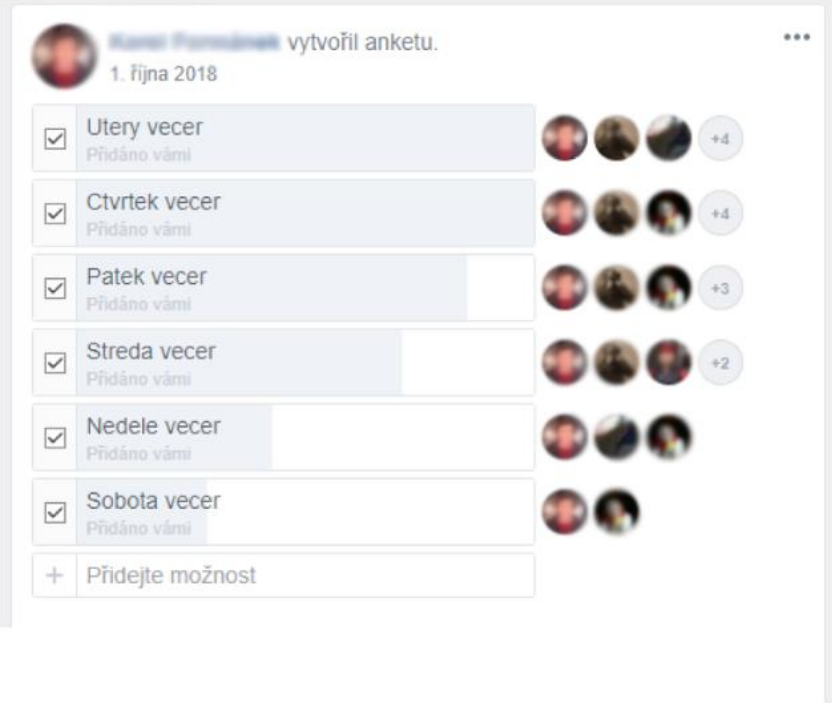
Schůzky

Schůzky probíhaly v nepravidelných intervalech jednou za dva až tři týdny o víkendu. Ve většině případů v neděli večer. Uskutečňovaly se přes online komunikační aplikaci Discord. Vždy v půlce týdne projekt manažer udělal anketu ve webové aplikaci Facebook, v jaký den a čas se schůzka uskuteční, na kterou měl každý člen přístup. Dle toho se vědelo, komu daný termín nejlépe vyhovuje a s kým se může případně počítat. Na základě ankety se všem oznámilo, kdy se schůzka uskuteční. Schůzky byly z velké části uskutečňovány proto, aby se členové od sebe sociálně nedistancovali.

Na schůzkách se probíraly organizační věci, které zahrnovaly například plánování účasti na herních veletrzích, kde se představovala rozpracovaná hra hráčské komunitě a herním médiím. Dále se probírali potencionální investoři, se kterými se společnost potýkala a také možnou budoucí krátkou neaktivitu některých členů ve vývoji z důvodu jejich osobních aktivit.

Při organizačních schůzkách se řešily problémy ve vývoji, například, zda každý správně pochopil vypracování jemu přidělené části daného projektu, jak jej popřípadě správně zhotovit nebo mu pomoci s jeho dokončením. Někteří členové organizačního týmu si záměrně nechávaly složitější vypracování zadaného úkolu až na schůzky, aby mohli daný problém prodiskutovat se všemi členy týmu a vybrat tak nejlepší variantu. Nebylo to tak pokaždé, neboť někteří nemohli z časových důvodů toto řešit s projekt manažerem.

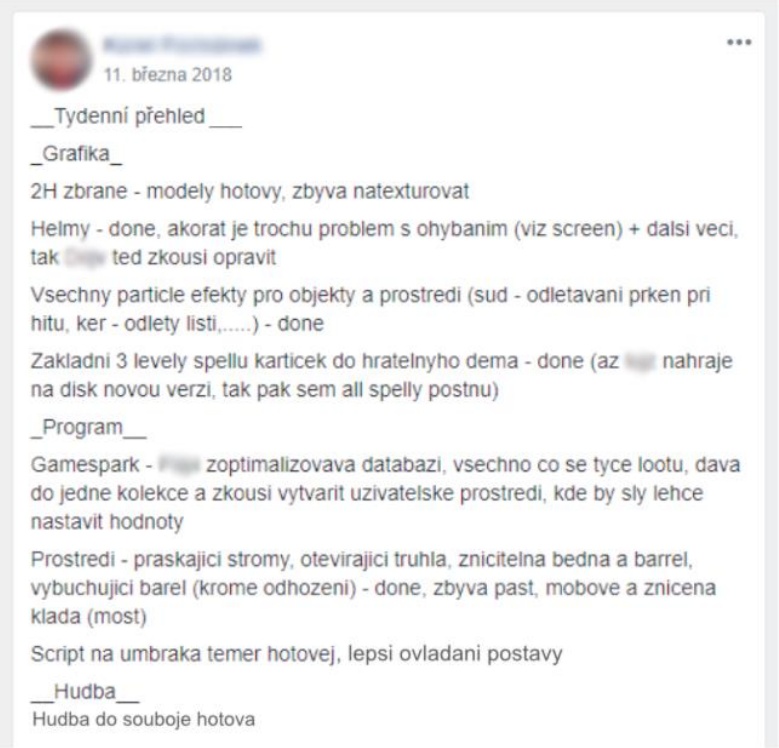
Poté projekt manažer každému přidělil nový úkol a vysvětlil mu, jak vše zamýšlel, ovšem jen za předpokladu, že dotyčný člen stále ještě neměl nějaký úkol rozpracován. Po přidělení úkolu skončila povinná účast a následovala již necílená diskuze, která neměla s projektem nic společného a bylo pouze pro sociální zlepšení vztahů týmu.



Obrázek 5 - Výstřižek ankety z roku 2018 (Zdroj: autor)

Týdenní souhrny

Mezi schůzkami byly na konci týdne týdenní souhrny, neboli reporty. Projekt manažer obepsal každého člena, jak s úkolem pokročil. Odpovědi všech členů vývojového týmu pak zpracoval a zveřejnil do společného příspěvku, aby všichni vědeli, jak se za tento týden vývoj poposunul kupředu. Report byl rozdělen do kategorií – Grafika, Program, Zvuk, Herní mechaniky. V každé kategorii byl vždy souhrn změn a pokroku.



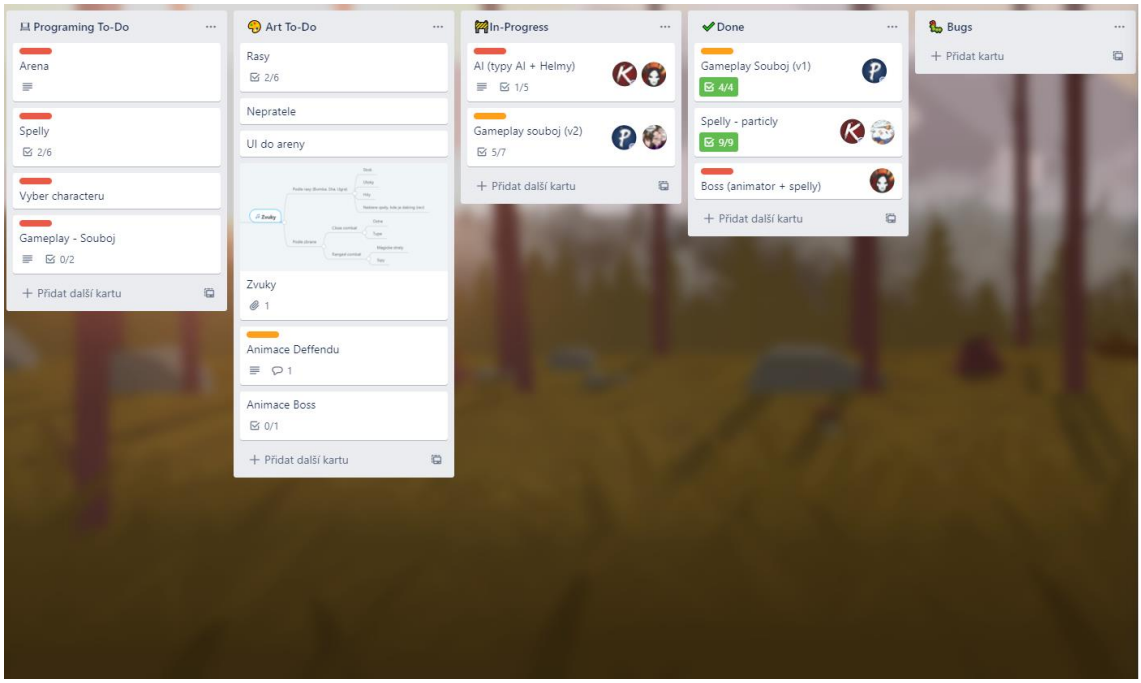
Obrázek 6 - Výstřižek jednoho reportu (Zdroj: autor)

3.3.2 Z hlediska řízení projektu, plánování a sledování práce na projektu

Projekt byl řízen přes webovou aplikaci Trello. Tato aplikace funguje na principu nástěnek, v každé nástěnce jsou sloupce, které obsahují lístečky neboli úkoly. Úkoly se přesouvají z jednoho slupce do druhého. Ke každému úkolu lze přiřadit profil člena týmu, doplnit popis, odškrtávací seznam a nahrát přílohu.

V případě tohoto herního projektu byly dva sloupce, které fungovaly jako backlogy. V jednom se sepisovaly úkoly programovacího rázu (Programming To-Do), ke kterým jsou potřeba programovací znalosti a v druhém uměleckého rázu (Art To-Do), ke kterým bylo potřeba grafických nebo animačních znalostí nebo práce s audiem.

Třetí sloupec (In-Progress) byl určen pro úkoly, na kterých se pracovalo. Do čtvrtého sloupce (Done) se přesouvaly úkoly ze třetího, které byly hotovy. V pátém, posledním sloupci se sepisovaly bugy, chyby v kódu, v projektu, neboli jakékoliv zvláštní anomálie objevující se v softwarovém projektu, které nefungují tak jak mají.



Obrázek 7 - Výstřižek z Trella, který zobrazuje projektové plánování (Zdroj: autor)

Úkoly vytvářel, přesouval mezi sloupci, mazal a přerozděloval členům pouze projekt manažer. Jak již bylo zmíněno v předchozí kapitole, sepsání úkolů do prvních dvou To-Do sloupců bylo na základě předsudku projekt manažera, co si myslel, že herní projekt bude všechno obsahovat. Chyběl jakýkoliv dokument, který by jasně vylíčil funkcionality herního projektu a jeho cíl. Vývojový tým věděl, o čem bude hra nebo co bude obsahovat pouze z ústních informací od projekt manažera, chyběla jakákoliv písemná forma, čeho by se dalo při vývoji držet.

Jednotlivé úkoly byly sepsány dle jednotlivých herních mechanik, to znamená například, že jeden úkol „Pohyb postavy“ obsahoval, jak se má postava ve hře ovládat, jak má skákat, jaké animace má mít, jaké efekty se mají při pohybu vytvářet a jaké zvuky se mají přehrávat. V podstatě jeden takový rozsáhlý úkol byl brán jako jeden celek, který se musí udělat.

Zde nastávaly dva výrazné problémy při sepsání úkolu.

1. Vytvoření a popis úkolu nebyl prodiskutován s nikým dotyčným v té profesní oblasti, tudíž celý úkol záležel pouze na znalostech projekt manažera, který ho sepisoval, takže se zde mohly objevit věci, které nejsou možné zrealizovat.
2. Některé úkoly byly velice rozsáhlé a také se počítalo, že vytvořené funkcionality v úkolu se budou brát za finální a objeví se ve hře.

Kvůli takto rozsáhlým úkolům, museli ti, co na nich dělají, někdy čekat na splnění a nahrání jiných věcí od ostatních členů týmu, s kterými musí ve svém úkolu pracovat a pak takto velký úkol zabral mnoho času k jeho dokončení – někdy i celé dva měsíce. Během toho času se v projektu změnilo tolik věcí, že funkcionality, které se udělaly v tom úkolu, se staly nepotřebnými.

Jeden z konkrétních příkladu byla například databáze pro hráčské účty, která měla obsahovat přihlašovací údaje uživatele, jeho vytvořenou postavičku a celý jeho hráčský pokrok ve hře spolu s inventářem, kde měl sesbírané herní předměty. Tvořila se takto velká databáze s tím, že se již do projektu použije. Tudíž po celou dobu vytváření, nebyla databáze dostupná k napojení na hru a k otestování. Když byla databáze z poloviny hotová, tak se tato online funkcionalita z projektu odstranila a tím pádem práce na databázi byla naprosto zbytečná.

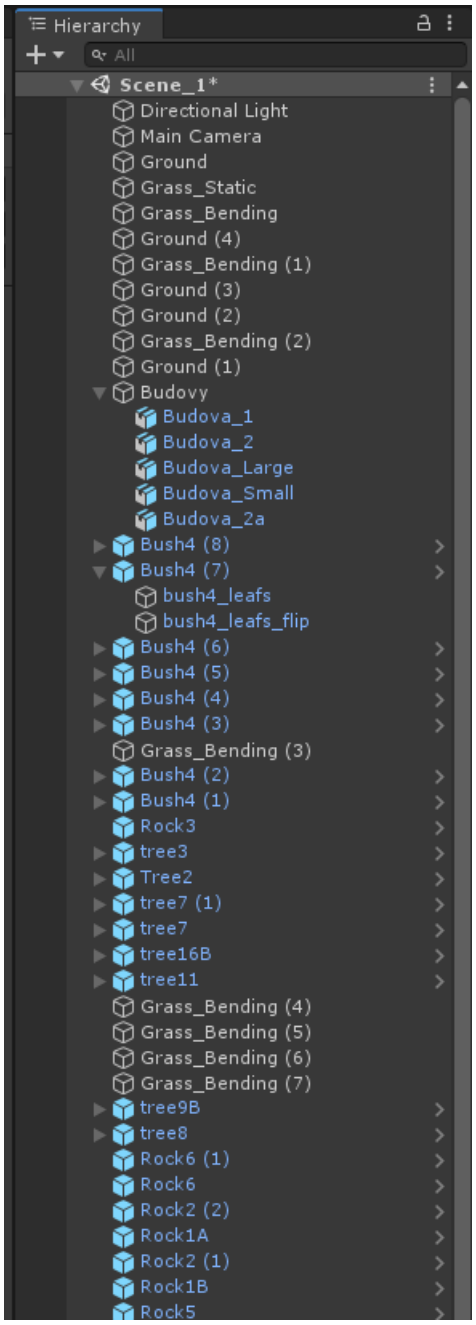
Druhý konkrétní příklad se jednal ohledně grafického obsahu 3D modelů zbraní a prostředí. Vymodelovat 180 zbraní trvalo zaokrouhleno celý měsíc. V průběhu tvorby těchto 3D modelů neprošlo žádné vizuální zkontrolování o tom, jak tyto 3D modely vypadají. Tudíž po dodělání těchto zbraní se tyto modely mohly od začátku předělat, jelikož svým vzhledem nezapadaly do grafické stylizace hry. S prostředím byl zas problém, že byla zbytečně vynaložená práce na tvorbu 3 různých prostředí – jehličnatý les, temný les, zimní prostředí, zatímco se pro přímo pro testování herních mechanik používal a pouze jehličnatý les a neustále se jeho design upravoval tak, aby vyhovoval herním mechanikám, takže zbylé prostředí se mohly nanovo předělat kvůli starému designu.

Projekt se neřídil pravidelnými iteracemi, kde by na konci nějakého období bylo hratelné demo. Hra byla spustitelná v herním enginu a nikdo, kdo přímo v enginu nepracoval, tak neměl možnost si ji vyzkoušet či otestovat. Tudíž vytváření nových úkolu

do backlogu, bylo zase na projekt manažerovi, co si myslel, že je potřeba dodělat a tím se opět projekt mohl dostávat do fáze, kdy se dělaly úkoly, které nebyly zrovna prioritní a mnohdy se tato práce posléze stala zbytečnou.

3.3.3 Z hlediska práce na projektu a práce se soubory

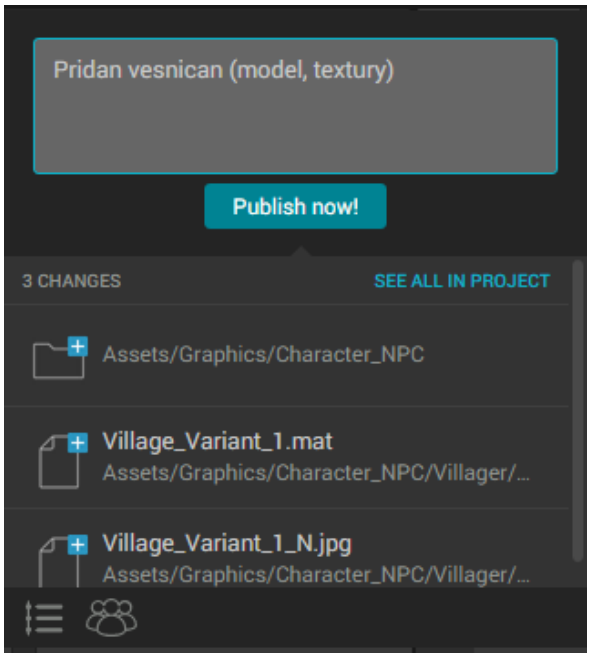
Herní projekt je vyvíjen v enginu Unity. Unity obsahuje adresář pro správu souborů v projektu a i okno se seznamem objektů umístěné v grafickém editoru. Bohužel chyběla pravidla pro strukturu adresářů v projektu Unity a i v grafickém editoru, jelikož si každý člen objekty a soubory v editoru skládal či pojmenovával podle sebe. Pak se tyto soubory složitě vyhledávaly a svým pojmenováním nešlo ani poznat, jaká textura, materiál či model k sobě má patřit. To samé v editoru, stávalo se, že se do editoru přetáhlo mnoho objektů bez jakéhokoliv roztřídění a pak kvůli tomu nastal naprosto nepřehledný chaos.



Obrázek 8 - Výstřižek okna Hierarchy, které zobrazuje objekty v 3D editoru (Zdroj: autor)

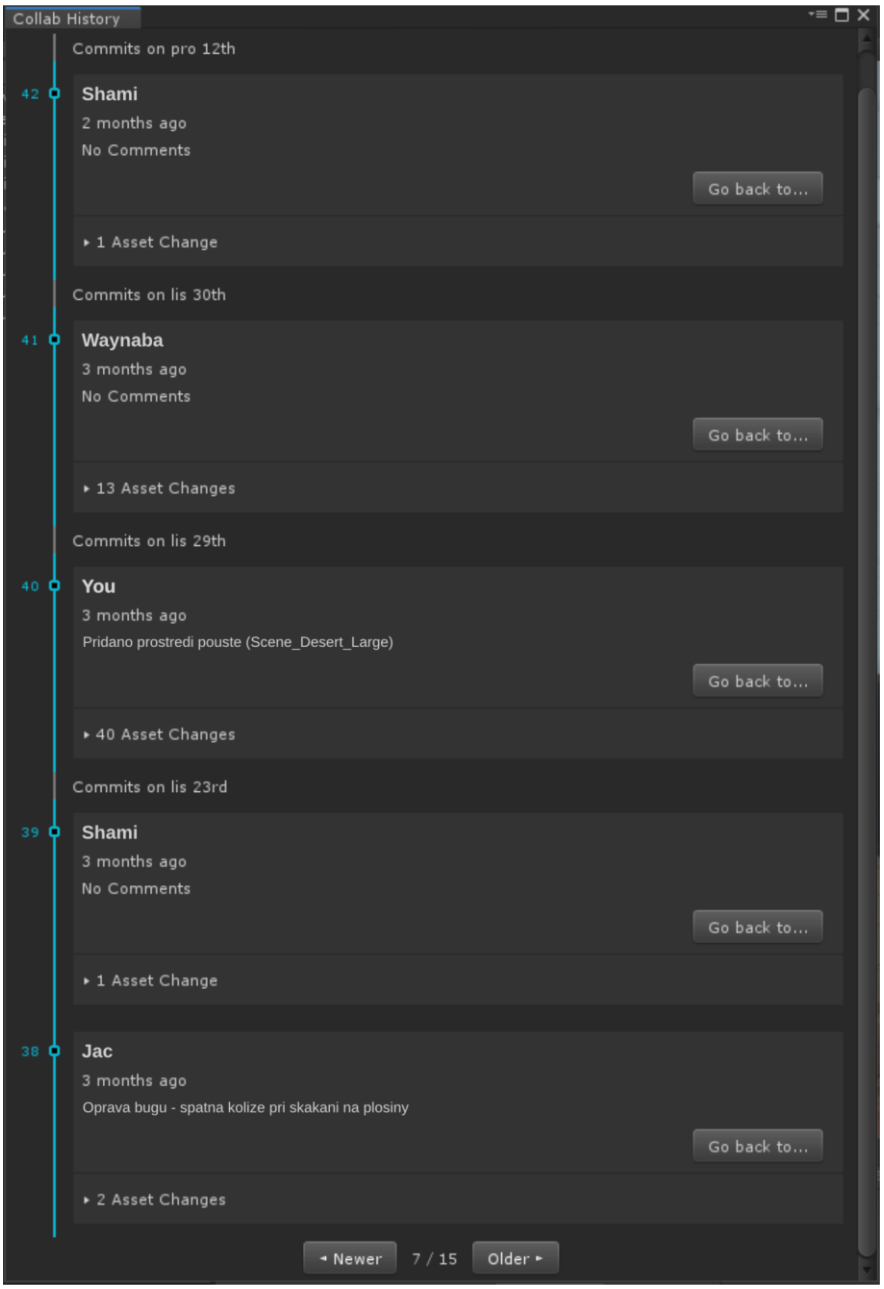
Z obrázku 8 jsou vidět objekty umístěné v Unity editoru, ovšem je zde tolik různých nesetříděných objektů, které tím ztěžují práci v samotném editoru. Například je zde hodně objektů, které zobrazují stromy, kameny, trávu, jenž by mohly být kategoricky roztříděné do nových prázdných objektů, díky kterým by se zvýšila přehlednost a nebyl by to jen dlouhý seznam náhodně po sobě seřazených objektů.

Soubory, které byly v adresáři projektu unity, byly verzovány přes službu Collaboration, která je nativně implementovaná v enginu Unity. Jedná se o obdobu Gitu. Collaboration si sám hlídá změny v projektu – přidání, smazání, přepsání souboru. Pak lze tyto změny nahrát. Ostatní členové si tyto změny stáhnou k sobě do Unity projektu a všichni tím pracují se stejnou aktuální kopií projektu.



Obrázek 9 - Výstřižek okna z Collaboration v unity při novém nahrání (Zdroj: autor)

Ovšem málokdo psal popis k nahraným změnám v Collaboration. Napsal je pouze do společného chatu nebo přímo projekt manažerovi. Takže buď ostatní nevěděli, co se zrovna aktuálně nahrálo do projektu nebo pak zpětně nešlo poznat, jaká změna co obsahovala.

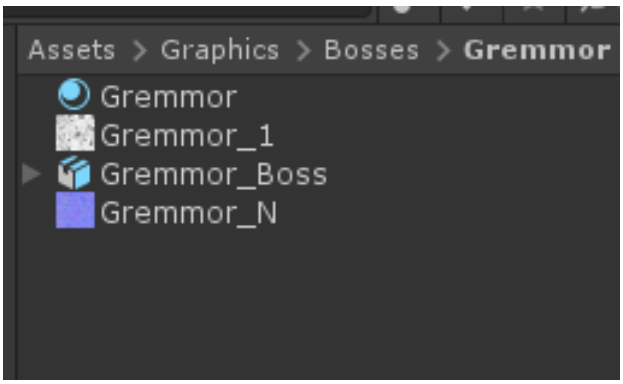


Obrázek 10 - Výstřižek nahraných verzí v Collaboration (Zdroj: autor)

Z obrázku 10 jsou vidět jednotlivé nahrané změny. Většina z nich má v popisu pouze „No Comments“, což se automaticky vyplní, pokud se do popisu při nahrání nic nevypíše, tudíž z toho nejde poznat, co se mohlo změnit, pokud si tedy nerozklikneme detaily a postupně si neprohlédneme soubor po souboru.

Soubory, které byly vytvořeny na základě jednotlivých úkolů a nevyžadovaly přímo práci v herním enginu, členové týmu nahrávali do sdílené složky na Google Disku. Jednalo se převážně o grafický obsah pro hru nebo audio. Skripty se vždy nahrávali přímo

v enginu přes službu Collaboration. Takto nahrané soubory se stejně museli přidat do projektu v Unity. Dělal to projekt manažer, pokud musel zkontrolovat daný soubor, nebo dotyčný člen týmu, pokud jeho práce vyžadovala práce s tímto souborem v Unity. Problémem bylo, že některé soubory byly pojmenované česky a některé anglicky, neměly konzistentní pojmenování a tím pádem se i špatně vyhledávaly. A to jak v adresářích na Googlu Disku, tak i adresářích v Unity.

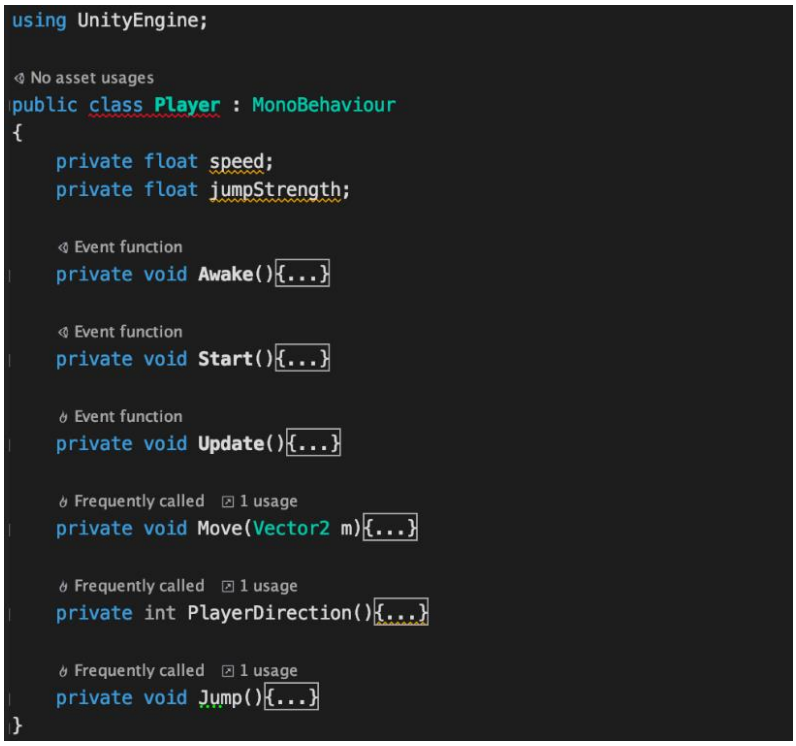


Obrázek 11 - Výstřižek pojmenovávání souborů (Zdroj: autor)

Na obrázku 11 v levé části jsou zobrazeny 4 soubory.

- Gremmor_Boss, což je 3D model s animacemi
- Gremmor, což je materiál
- Gremmor_1, což je difúzní textura
- Gremmo_N, což je normálová textura

Není zde žádný systém v pojmenovávání jednotlivých souborů a kvůli tomu, je pak obtížné jejich vyhledávání, manipulace s nimi a na první pohled se ani nedá poznat, o co se jedná.



Obrázek 12 - Výstřižek kódu bez namespace (Zdroj: autor)

Velkým problémem u psaní kódu ve vývojovém týmu byla absence v používání „Namespace“ (jmenný prostor), který pomáhá k lepší organizaci skriptu. To znamená, že pokud více jak jeden programátor pojmenuje svoji třídu stejným názvem, například „Player“ jako je v obrázku 12, tak dojde ke kolizi. Pokud by se třída použila v namespacu, tak může být použito více tříd stejného názvu v projektu.

Dále zde nebyla žádná ucelená pravidla pro programování v herním enginu Unity. Každý programátor v týmu si je psal dle svého, což neprospívalo týmové práci a bylo obtížné se v kódu vyznat.

3.3.4 SWOT analýza vývojového týmu

Silné stránky	• Velmi motivovaní členové v týmu • Lidé v týmu se chtějí naučit nové věci a mají přehled o herních produktech • Přibližně stejná věková skupina lidí v týmu, velmi podobné zájmy.
Slabé stránky	• Nedostatek zkušeností s herním vývojem, vedením týmu a projektováním. • Špatná komunikace v týmu při předávání informací • Ne všichni na tom pracují „full-time“ • Některým členům týmu chybí výrazné zkušenosti s určitými programy, které se používají pro herní vývoj • Občas dosti odlišné názory, které se silně prosazují a neprospívají týmu • Neefektivní postup vývoje
Příležitosti	• Nabídky od potencionálních investorů • Kontakty na lidi z herního průmyslu a herních médií • Rozjetý herní trh a velká komunita lidí, tudíž si dost vývojových věcí v oblasti programování a tvorby grafiky lze dohledat • Zkušenosti z vlastního vývoje
Hrozby	• Kvůli špatné komunikaci a nedostatku času členů týmu se často lze dostat do situace, kdy nebude správně rozložena pracovní síla • Demotivace z nekonečného vývoje a absence hratelných ukázek k vyzkoušení

Tabulka 3 - SWOT analýza (Zdroj: autor)

3.3.5 Souhrn starého postupu vývoje

Starý vývoj popsaný v bodech:

1. Sepsání seznamu úkolů projekt manažerem do webové aplikace Trello do sloupce s názvem To-Do. Názvy úkolů mají odpovídat tomu, o co v úkolu půjde. Případně mají obsahovat zaškrtávací seznam, pokud úkol obsahuje více položek ke splnění.
2. Projekt manažer po konzultaci, uskutečněnou přes online komunikaci, vysvětlil a přidělil členu týmu jeho úkol ze seznamu úkolu nebo ze seznamu chyb (bug list), na kterém bude pracovat. Také od něj zjistil jeho odhad, jak dlouho mu splnění zabere. Tento časový odhad se zapsal do Trella jako konečný termín splnění úkolu. Tudíž se předpokládá, že do konečného termínu bude úkol hotov. K úkolu se také přidá Trello profil člena týmu pro přehlednost, aby každý věděl, na čem má pracovat a věděl, na čem pracují ostatní členové. Takto přiřazený úkol se přesune ze sloupce To-do do sloupce In-Progress.
3. Pokud někdo narazí na nějakou chybu (bug). Tím se myslí něco, co nefunguje dle zadaného úkolu, napíše se do seznamu chyb (bug list).
4. Pokud někoho napadne nějaké vylepšení, které se týká herních mechanik, grafické stránky, či kódu. Napíše ho do nově vytvořeného úkolu do sloupce IDEAS. Pak je na projekt manažerovi, aby tento nápad zvážil a někomu přidělil.
5. Když se úkol splní, tak ho člen týmu, který ho právě dodělal, přesune do sloupce DONE a oznámí to projekt manažerovi. Aby se vědělo, co je splněno.
6. Na konci každého týdne se uskuteční report, kdy projekt manažer obepíše všechny členy týmu, co za tento týden zvládli udělat. Pak na základě jejich odpovědí, udělá soupis změn a přidaných věcí. Tento report se zveřejnil v soukromé chatovací skupině, kde byl každý, co se podílel na vývoji.

3.3.6 Kritická místa

V tomto postupu je hned několik kritických míst, díky kterým se nedosahuje viditelných výsledku.

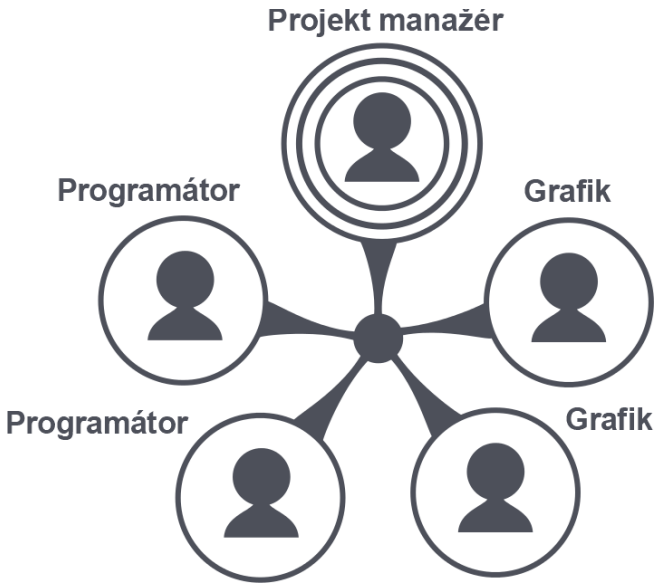
- Chybí game design dokument, tudíž chybí pořádně sepsaný cíl a funkcionality hry, které se mají dosáhnout. V podstatě bez toho je to nekonečný vývoj herního softwaru a zároveň v tomto případě projekt manažer nemá z čeho vycházet, při

psaní jednotlivých úkolu do backlogu (seznamu úkolů). Tyto backlogy sepisoval na základě toho, co si myslel, že má herní projekt obsahovat.

- Chybí potřebný smysluplný popis k úkolu, aby bylo jasné, proč je vůbec tento úkol zařazen v seznamu a jakou hodnotu přinese jeho splnění a co přesně má obsahovat.
- Velmi chybí nějaký systém, na základě, kterého se budou rozdělovat úkoly členům, aby se nestávalo, že se dělají nepotřebné věci, jelikož celý vývojový tým spoléhal na to, že projekt manažer vše dobře naplánoval a přerozdělil, nikdo se o nic víc nestaral a ani do ničeho už nezasahoval. Tomu také přidával fakt, že nikdo si neodzkoušel žádnou hratelnou verzi hry, protože se ani žádné hratelné verze nedělaly. Celý herní projekt byl pouze stále rozpracován v herním enginu, neboli v programu, který je určen pro tvorbu herního softwaru. Tudíž ten, kdo v něm přímo nepracoval, tak neměl ani možnost si nové funkcionality vyzkoušet.
- Problém s vývojem složitých úkolů. Úkoly byly přiřazeny projekt manažerem členům a počítalo se dopředu s tím, že to daný člen zvládne a chybělo jakékoliv prodiskutování, jak by se tento úkol mohl nejlépe udělat. V podstatě dotyčný člen byl na tento úkol úplně sám, pokud si sám nesehnal pomoc od odbornějšího člena. Chybí nějaký jednoduchý přehled typu úkolů na základě potřebných vývojových dovedností, aby se vědělo, zda se jedná o úkol, ke kterému jsou potřeba programátorské dovednosti nebo grafické dovednosti nebo dovednosti pro úpravu audia.
- Problém s touto prací na dálku spočíval v tom, že ne každý týdenní reporty četl a bylo také obtížné zajistit, aby je každý řádně četl. Velkou pravděpodobností to bylo způsobeno, že reporty se vypsaly o víkendech, kde tomu nikdo nevěnoval pozornost. Další problém byl, že report byl závislý pouze na online zprávě a na tom, jak detailně a smysluplně to každý člen do zprávy napsal. Stávalo se, že někdy o víkendech si ani někteří členové zprávu od projekt manažera ohledně jejich postupu práce nepřečetli.
- Chybí programátorská pravidla pro psaní kódu, pojmenovávání tříd, metod, proměnných, konstant atd. Jelikož dosud si každý píše kódy dle sebe a kvůli tomu, se kódy stávají velmi nepřehlednými.

3.4 Návrh nové metodiky

Z předešlých důvodů, je potřeba si sepsat kritéria nové metodiky, kterých je třeba docílit pro dosažení lepších výsledků ve vývoji. Zároveň v době zavádění nové metodiky, vývojový tým už začínal vyvíjet i aplikace ve virtuální realitě (VR) na zakázku pro nově založenou dceřinou společnost, což přineslo nové finanční prostředky a také z toho důvodu bylo potřeba upravit stávající složení týmu.



Obrázek 13 - Složení nového vývojového týmu (Zdroj: autor)

Kvůli vývoji pro VR a finančním prostředkům vzhledem k efektivitě vývoje a také k okolnostem investorů se nyní vývojový tým skládá z 5 členů. Tento tým nemá pouze fungovat na klasické hierarchii nadřízený/podřízený, ale má podporovat samo-organizující tým. Projektový manažer v tomto případě nefunguje pouze tak, že sleduje práci na projektu a přerozděluje úkoly, ale má i vybízet ostatní členy, aby se také zapojili do naplánování projektu, což se dříve nedělo.

3.4.1 Kritéria pro metodiku

- **Vzdálená práce**

Metodiku je třeba přizpůsobit pro práci na dálku, jelikož nebylo možné, aby všichni členové neustále byli spolu v jedné místnosti po většinu času.

- **Minimum dokumentace**

Potřeba zařídit, aby členové týmu neztráceli zbytečně čas psaním rozsáhlých dokumentací, které s největší pravděpodobností by nikdo pořádně ani nepřečetl. Bylo potřeba vytvořit nějaká pravidla pro to, co je potřeba dokumentovat a co není a jak to dokumentovat.

- **Všichni musí mít přehled o tom, co každý dělá a musí jim to dávat smysl**

Tohle kritérium má mentálně podporovat práci v týmu a případnou svižnou komunikaci při řešení problémů a hlavně podporovat motivaci při vývoji, když každý ví, na čem zrovna pracuje jeho kolega a může se těšit při skončení sprintu na další dokončené funkcionality a ne pouze slepě dělat přidělené úkoly.

- **Přehlednost**

Jeden z velice důležitých bodů kritéria, aby se z řízení projektu a také z práce se soubory, jak grafickými, zvukovými či skripty, nestal nepřehledný zmatek, kde by každý člen zdlouhavě zjišťoval, na jakém úkolu má dělat a s jakými soubory má zrovna pracovat anebo v jakém adresáři má najít potřebné soubory.

- **Viditelné výsledky**

Hlavním důvodem je především předejít demotivování tím, že členové týmu uvidí, že se v projektu pokračuje a opravdu se objevují nové funkcionality.

- **Práce se soubory musí být přizpůsobena pro týmovou práci**

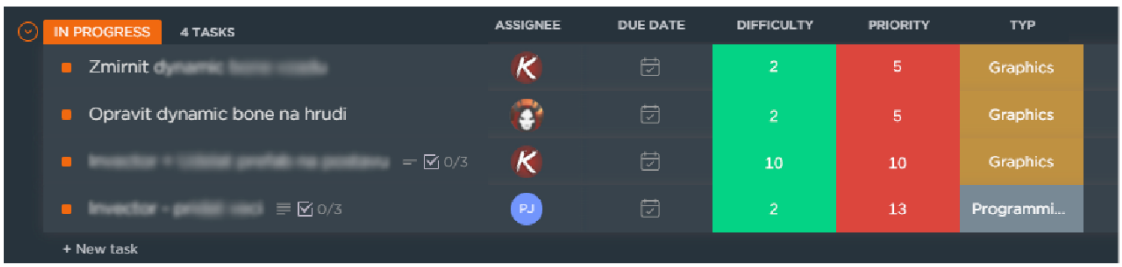
Soubory a jejich změny musí být dobře a rychle vyhledatelné. Musí být zároveň ukládané ve formátech, které každý člen týmu jednoduše otevře a jejich pojmenování musí být logické vzhledem k danému souboru.

3.4.2 Body nové metodiky

Na základě kritérií a předchozí analýzy je sepsán návrh nové agilní metodiky v následujících bodech. Z důvodu některých chybějících funkcionalit webové aplikace Trello, se projekt řídí přes aplikaci ClickUp, která je pro tuto metodiku mnohem vhodnější. Aplikace ClickUp je podrobněji rozebrána v kapitole 2.6 (Nástroje pro podporu a vývoje metodiky). Cíl této metodiky je hlavně podpořit práci a komunikaci v týmu a zařídit, aby se každý člen plně zapojil do vývoje.

1. Sepsání Game Design dokumentu, který má dát celkový přehled o hře.
    - Analýza hry – popsat základní informace o hře, její cíl, čeho má hráč dosáhnout, co bude překonávat. Žánr hry. Pro jakou cílovou skupinu hráčů je hra určena.
    - Na jaké platformy bude hra vyvíjena.
    - Příběh hry - do jakého světa je hra zasažena. Jaké postavy se budou zde vyskytovat a jejich charakteristika.
    - Popsat hratelnost, jak se bude hra ovládat, jaké budou možné interakce ve hře, jaké budou herní módy, jaký zážitek hra nabídne hráčům. Co bude hráče dál motivovat, jaké odměny bude moci dostat.
    - Level design – jaké scény, levely, úrovně bude hra obsahovat, co se v každé scéně bude vyskytovat.
    - Grafická stylizace – jak budou jednotlivé objekty, modely, postavy vypadat. Jaké bude uživatelské rozhraní.
2. Na základě Game Design dokumentu společně vývojový tým sepíše celý backlog, neboli seznam veškerého obsahu a funkcionalit, co má hra obsahovat. Vývojovým týmem se myslí lidé, kteří se budou aktivně podílet na vývoji hry.
3. Nastaví se jednotýdenní vývojové iterace, neboli sprinty, které byly převzaty z agilní metodiky SCRUM. V tomto týdnu je potřeba dokončit všechny přiřazené úkoly. Na začátku tohoto období, v pondělí, se uskuteční fyzická schůze všech členů týmu, kde se předvedou a zkontrolují hotové úkoly z předešlého sprintu.

4. Zároveň se týmově naplánuje další sprint a přiřadí úkoly. Zbylé pracovní dny se již pracuje vzdáleně, to znamená, že není potřeba fyzická účast v jedné místnosti v kanceláři.
5. Každý den proběhnou krátké reporty kromě pondělí. Každý člen týmu napíše do společného online chatu, na čem včera pracoval a co má dokončené. Tyto reporty mají za cíl mentálně podporovat motivaci na vývoji.
6. Jednotlivé úkoly se ohodnotí dvěma hodnoceními - důležitostí a náročností. Ohodnocovat bude celý vývojový tým na základě toho, co je zrovna v aktuálním vývojovém týdnu nejdůležitější udělat, vzhledem k současnému stavu projektu. Na hodnocení důležitosti a náročnosti se použije Fiboancciho posloupnost (1,2,3,5,8,13). Nejdůležitější úkol anebo nejnáročnější se ohodnotí číslem 13. Ten nejméně důležitý anebo nejlehčí se ohodnotí 1. Zároveň musí mít popis, který bude dávat smysl, proč se má daná funkcionalita přidat a jakou hodnotu přinese. Tento popis má sloužit k ujasnění úkolu, jestli má vůbec smysl se něčím takovým zabývat. Největší priorita se bude přiřazovat v pořadí úkolům, které mají potenciál co nejvíce podpořit a obohatit hratelnost hry. K úkolů se také kvůli přehlednosti přidá označení, dle toho, zda se bude jednat o práci pro grafiky, programátory nebo práce na Game Designu, což je vymyšlení a sepsání herních mechanik.

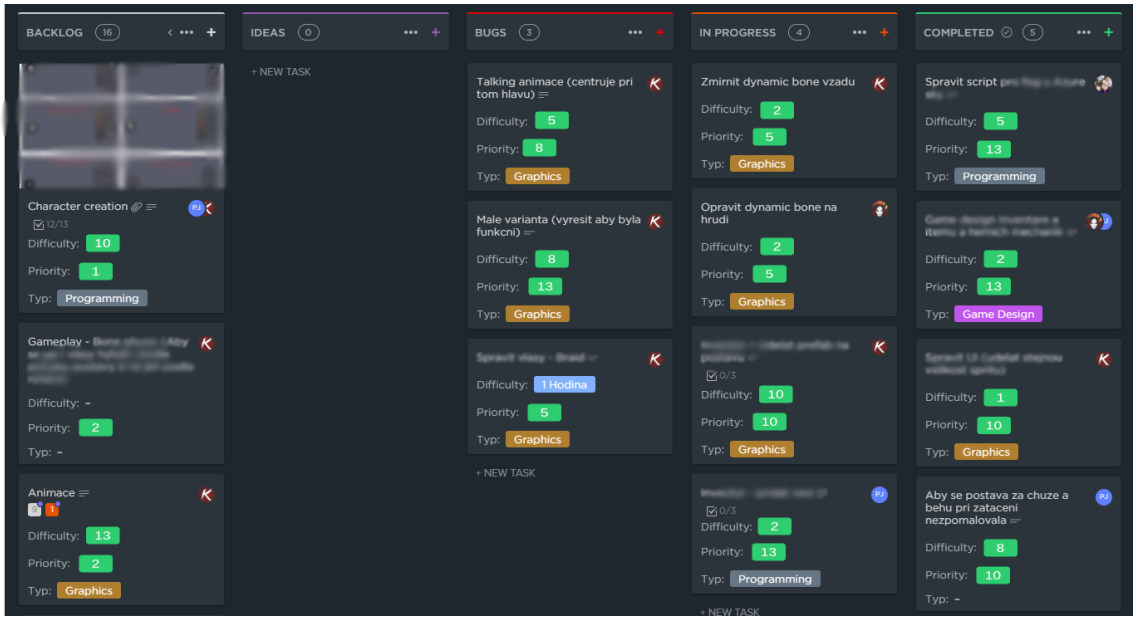


Obrázek 14 - Ukázka několika úkolů a jejich ohodnocení (Zdroj: autor)

7. Na základě ohodnocení se dané úkoly přiřadí ke členům týmu, s tím, že se v tomto sprintu dokončí a budou se moci ukázat a případně probrat na pondělní schůzi. Přiřazovat se bude na základě důležitosti úkolu vzhledem k jeho náročnosti a zda má smysl aktuální funkcionality v úkolu dokončit v následujícím sprintu. Při pondělních schůzích se vždy projde celý backlog,

jestli nemá smysl přerozdělit jednotlivým úkolům jejich důležitost, která se může již hotovými funkcionalitami měnit a také se může stát, že některé funkcionality v backlogu již nebudou potřeba a některé se naopak mohou stát velmi důležitými.

8. Sepsané nepřiřazené úkoly budou ve sloupci Backlog. Přiřazené úkoly budou ve sloupci In-Progress. Jakékoliv chyby, které je potřeba opravit budou ve sloupci Bugs. Úkoly, které se dokončí, se přesunou do sloupce Completed. Po předvedení a zkontrolovaní úkolu celým týmem, se tento úkol přesune do sloupce Closed. Jakmile někdo objeví jakoukoliv chybu, ihned ji napíše jako nový úkol do sloupce Bugs. Při pondělních schůzkách se tyto Bugy proberou. Pokud někoho napadne nová mechanika nebo dostane nápad, napíše ho do sloupce Ideas. Při schůzkách se tyto nápady také proberou, zda mají smysl, aby se z nich stal plnohodnotný úkol do backlogu.



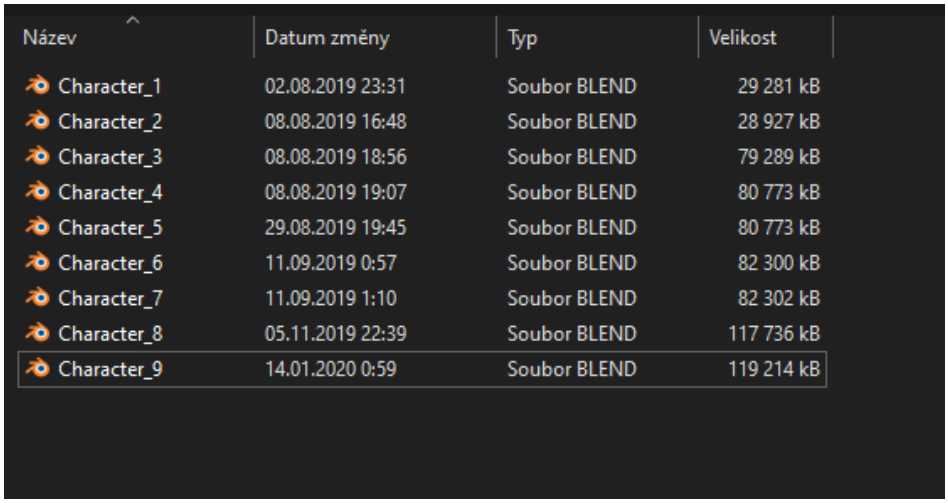
Obrázek 15 - Výstřižek z aplikace ClickUp, určená pro plánování projektu, při nově zavedené metodice (Zdroj: autor)

3.4.3 Nová pravidla pro práci se soubory a pojmenováváním na projektu

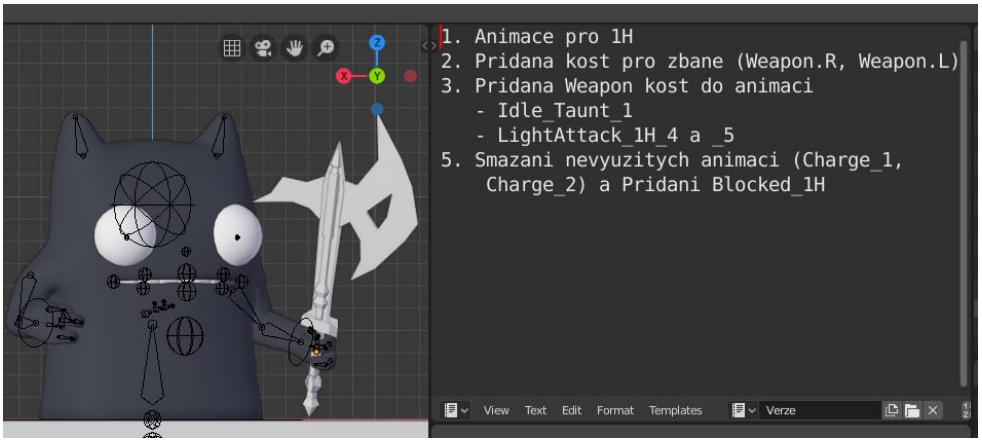
Z důvodu přehlednosti, jednoduší práci na projektu a vývojovým chybám v předchozí analýze se zavedou tyto pravidla.

3.4.3.1 Pravidla pro soubory

1. Verzovat soubory – každé změny ukládat do nové kopie souboru. Na konci názvu nové kopie souboru, napsat její verzi. Například, pokud se bude jednat o 3D model znázorňující kámen, tak se první verze modelu pojmenuje „Kamen_1“. Druhá verze se zas pojmenuje „Kamen_2“.



Obrázek 16 - Ukázka číslování verzí souboru (Zdroj: autor)



Obrázek 17 - Ukázka pojmenovávání změn ve verzích v Blenderu (Zdroj: autor)

2. Změny provedené v programu Blender, což je 3D grafický program, psát do Blender Text Editor k aktuální verzi souboru.

3. Textury ke grafickým modelům v Blenderu rovnou nahrát do souboru Blenderu, aby se k nim nemusela pokaždé hledat cesta. Zvětší to sice velikost souboru, ale za cenu opakovaně zbytečného přiřazování textur k materiálům.
4. Typ souboru musí mít na konci svého názvu svoji zkratku, aby bylo zjednodušena manipulace a vyhledávání s nimi
    - Difúzní textury = _D
    - Albedo textury = _A
    - Normálové textury = _N
    - Ambient occlusion, neboli dodatečné stínování = _AO
    - Specular map, neboli textura na lesk = _S
    - Metalická mapa = _M
    - Height map, neboli textura ke generování 3D povrchu = _H
    - 3D modely = _Model
    - Materiály = _Mat

Obrázek 18 - Příklad pro pojmenovávání souborů před zavedení pravidel (vlevo) a po zavedení pravidel (vpravo) (Zdroj: autor)

3.4.3.2 Pravidla pro práci v enginu Unity

1. Povinné popisy v Collaboration nahraných změnách, ale tak aby dávaly smysl, co se přidalo, co se změnilo, co se smazalo a proč.

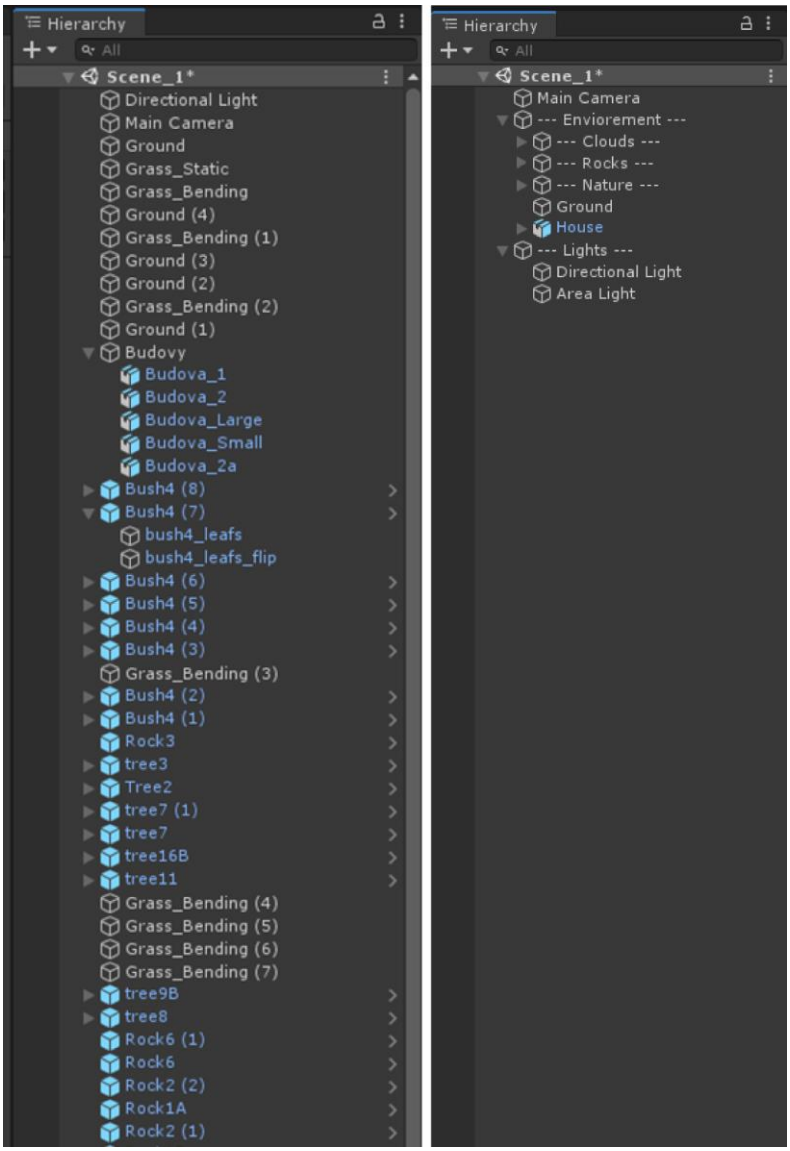
2. Každý člen týmu, co se podílí na vývoji, si vytvoří svoji pracovní složku v hlavním adresáři unity (složka Assets), kde na konci názvu složky bude jeho

jméno a na začátku podtržítko, aby složky byly řazeny na prvních místech. Např.: _Workplace_Karel

3. Organizovaná struktura kořenového adresáře (Assets) v Unity bude tímto způsobem:
    - Animations
        - (Zde budou soubory typu animace, kontrolér pro animace a masky)
    - Audio
        - Music
        - Ambient
        - Sound_Effects
    - Graphics
        - Environment
            - (Složky nějakého objektu)
                - 3D model, textury, materiál
        - HDRI
        - Characters
            - (obdobné jako u Environment)
        - Particle effects
        - Terrain
            - Ground_Textures
        - UI
            - Fonts
        - (Další adresáře kategoricky vytvořené)
    - Prefabs
    - Plugins
        - (Zde budou další složky obsahující potřebné assety třetích stran)
    - Scenes
    - Scripts
    - Shaders

3.4.3.3 Organizování v okně Hierarchy

Objekty ve scéně se musí třídit do nově vytvořených prázdných objektů a pojmenovat je kategoricky podle toho, jaké složky obsahují. Označí se třemi pomlčkami před a za názvem. To znamená, že všechna světla ve scéně budou v novém objektu pojmenovaném „--- Lights ---“. Statické objekty prostředí zase v objektu „--- Enviorement ---“. Pokud bude více duplikátů nějakých objektů, tak se přesunou do jednoho prázdného objektu s takovým pojmenováním, aby bylo jasné, co obsahuje.



Obrázek 19 - Příklad pro pojmenovávání objektů v okně Hierarchy před zavedení pravidel (vlevo) a po zavedení pravidel (vpravo) (Zdroj: autor)

3.4.3.4 Pravidla pro programování v Unity v týmu

1. **Pojmenovávání tříd**
    - Pojmenovávání pomocí „PascalCase“, což znamená, že slova začínají velkými písmeny a jsou bez mezer.
    - Název třídy musí být podstatné jméno

Příklad: `public class CharacterPlayer : MonoBehaviour {}`

2. **Pojmenovávání vlastností**
    - Pojmenovávání pomocí PascalCase
    - Musí být podstatné jméno nebo přídavné jméno

Příklad:
```
public class CharacterPlayer : MonoBehaviour
{
    Public int race { get; set; }
}
```

3. **Pojmenovávání proměnných**
    - Pojmenovávání pomocí velbloudí notace, což znamená, že první slovo začíná malým písmenem a následující zase velkým.
    - Musí být podstatné jméno nebo přídavné jméno
    - Název musí začínat podtržítkem

Příklad:
```
public class CharacterPlayer : MonoBehaviour
{
    private int _health;
}
```

4. **Pojmenovávání metod**
    - Pojmenovávání pomocí PascalCase
    - Musí být sloveso

Příklad:
```
public class CharacterPlayer : MonoBehaviour
{
```

```
        private int AttackSword() {}
    }
```

5. **Pojmenovávání parametru metody**
    - Pojmenovávání pomocí velbloudí notace
    - Musí být podstatné jméno nebo přídavné jméno

Příklad:
```
public class CharacterPlayer : MonoBehaviour
    {
        private int AttackSword(int value) {}
    }
```

6. **Pojmenovávání konstant**
    - Celý název velkými písmeny
    - Musí být podstatné jméno nebo přídavné jméno

Příklad:
```
public class CharacterPlayer : MonoBehaviour
    {
        public const int MAX_ITEMS = 10;
    }
```

7. **Pojmenovávání eventů (událostí)**
    - Použít PascalCase
    - Musí značit pojem „když se něco zrovna stane“. Například: něco se stane, když zrovna konkrétní objekt přejde z jedné animace do druhé.

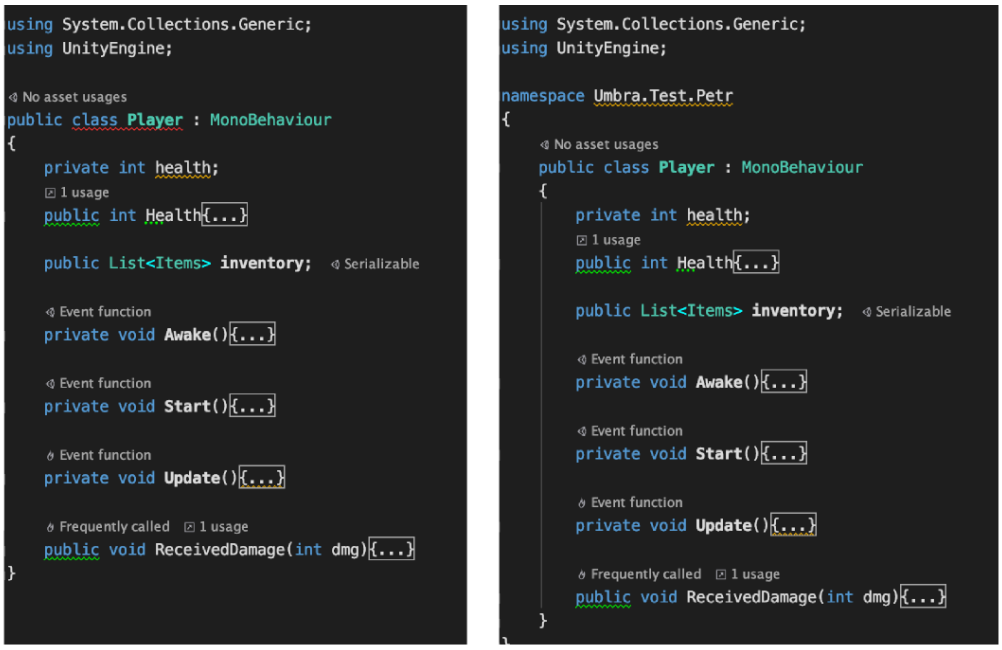
Příklad:
```
public class CharacterPlayer : MonoBehaviour
    {
        public event Action<int> OnChangeState;
    }
```

8. **Pojmenovávání zkratek**
    - Vyhnout se pojmenovávání názvů ve zkratkách s výjimkou běžného názvosloví.

Příklad: Raději používat **characterRace** než **cRace.** A spíše použít zkratku **CPU** než **CentralProcessingUnit**

9. **Používání namespace (obory názvů)**

Používat namespace při vytváření tříd k testování či prototypování herní mechaniky, aby nedocházelo ke zbytečné kolizi. To znamená, že se celá třída vloží do namespacu. Viz Obrázek 20.

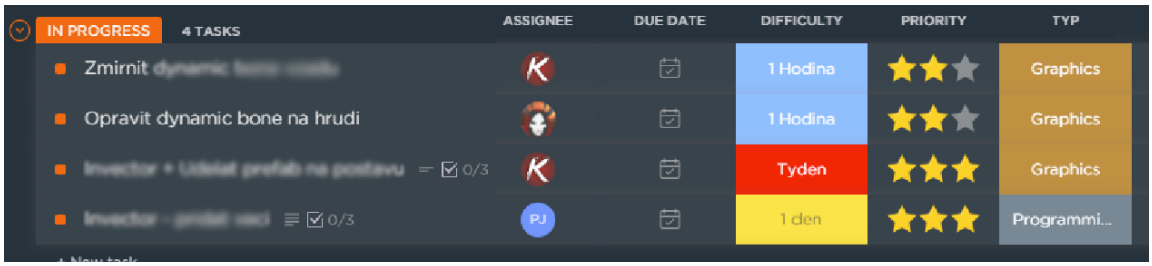


Obrázek 20 - Příklad pro používání namespace. Vlevo příklad bez namespace. Vpravo příklad, kdy je namespace použit (Zdroj: autor)

3.5 Vyhodnocení zavedení nové metodiky

Prvotním problémem za čtrnáct dní po zavedení metodiky bylo hodnocení úkolů dle Fiboancciho čísel, jelikož nedávaly smysl v obtížnosti a ani se podle toho nedalo dobře určit, kolik úkolů by mohl daný člen v týmu dokončit v jednom týdnu. Proto se ohodnocení jednotlivých úkolů z čísel (1,2,5,8,10,13) změnilo na ohodnocení podle

časové náročnosti (1h, 4h, 1 den, 3 dny, týden) a hodnocení důležitosti se zjednodušilo na počet hvězdiček, kde nula je nejnižší priorita a tři nejvyšší.



Obrázek 21 - Ukázka nového ohodnocení úkolů (Zdroj: autor)

Je zajímavé, že při společném plánování se ihned obtížnost u některých úkolů změnila z čísla 3 na jednu hodinu. Díky tomu šlo o dost lépe rozvrhnout a naplánovat pracovní týden. Dále také zjednodušení priorit z čísel na tři hvězdičky se lépe roztřídily úkoly, které jsou důležitější pro vývoj na zrovna aktuálním stavu, v jakém se projekt nachází. To znamená, že velmi důležitým úkolům se přiřadily tři hvězdičky a nepodstatným úkolům se nedala hvězdička žádná.

Krátké týdenní pracovní iterace s pravidelnou fyzickou účastí na pondělních schůzkách, kde se společně ukázalo, co se za týden udělalo a naplánovalo na nový týden, bylo velkým přínosem ve vývoji a na týmové souhře. Členové se celkově i podíleli na vymýšlení herních mechanik, které by mohly být pro hru zábavné či užitečné a hlavně si zde tvořili i citovou vazbu k projektu, že je to i jejich produkt.

Krátké denní reporty po prvních pár týdnech se moc neosvědčily a ne každý na ně reagoval, jelikož při časté online komunikaci stejně každý věděl, na čem kdo dělá a jak mu to jde, protože při nějakém problému se ihned diskutovalo. A zejména u grafiky, se ihned do společného chatu posílaly obrázkové ukázky o průběhu práce.

Pravidla třídění a ukládání dost pomohla k vyhledávání souborů a svižnější práci. Z delšího pozorování se zjistilo, že grafici vyhledávají soubory přímo v adresářích, takže správné roztřídění a pojmenování tu bylo přínosem a programátoři jednotlivé soubory vyhledávají ve vyhledávači. Ovšem zprvu byl problém, jelikož stále přetrvávaly problémy s česko-anglickými názvy (například „Tree_Vysoký“ – kdy část je anglicky a část česky) a občas velmi podobné názvy typu „Rocks“ a „Stones“, ale pojmenování u

textur, materiálů, modelů a animací bylo bezproblémové, stejně jako lepší organizace objektů v hierarchy okně v Unity.

V podstatě díky jednomu dokumentu, kde byla sepsána tato pravidla, se při dodržování náramně zpříjemnila a zefektivnila práce.

Zároveň díky společnému plánování, měl možnost celý tým ovlivňovat vývoj a díky pravidelným fyzickým schůzkám, se hodně zlepšily týmové vztahy členů, kteří si vytvořili k projektu citovou vazbu. Důsledkem této změny v organizaci práce se projevilo, že se více diskutovalo o návrzích k novým funkcionalitám a obsahu hry.

Průběh zavádění nové metodiky vývojový tým přijímal kladně, jelikož členové sami věděli, že je potřeba nějaká změna ve fungování, aby se dosáhlo lepších výsledků.

Shrnutí	
Starý postup vývoje bez metodiky	**Nová metodika**
Designování hry a psaní backlogu dle toho, jak si ji představoval projektový manažer.	Sepsání Game Design dokumentu, který ujasnil cíle hry i cíle celého projektu a na základě toho se sepsal celý backlog.
Přiřazování úkolů volným členům, nehledě na to, zda má zrovna smysl na dané funckionalitě pracovat.	Přiřazení obtížnosti a časové náročnosti k úkolům, což vede k efektivnějšímu naplánování týdne a zároveň se pracuje na aktuálně důležitých funkcionalitách, tudíž se neplýtvají prostředky na nepotřebné věci.
Počítalo se s tím, že daná funkcionalita v backlogu bude finální a objeví se ve finální hře.	Při každé společné schůzce, se projede celý backlog a u některých úkolů se může změnit jejich priorita. To vede k tomu, že některé funkcionality ve finále nebudou vůbec potřeba.
Celý týmový projekt byl řízen jedním člověkem, tudíž ostatní členové byli jen	Určování priorit a naplánování týdne celým týmem. Tým se stává samo-

od toho, aby pracovali na funckionalitách a nebyly pocitově spjati s projektem, že ho také můžou designovat.	organizující a má k němu větší citovou vazbu.
Schůzky probíhaly nepravidelně jednou za 2 až 3 týdny online formou. Celé je vedl projektový manažer. Členové byli jen pasivními posluchači.	Pravidelné fyzické schůzky jednou týdně, při kterých se členové aktivně zapojovali do společného naplánování vývoje.
Týdenní souhrn vývoje, se zveřejnil online. Bylo obtížné zajistit, aby si ho každý přečetl a měl díky tomu přehled o vývoji.	Při schůzkách se všem ukázalo, co každý za ten týden udělal a prokonzultovali se případné zlepšení nebo problémy. Hlavně každý má dostatečný přehled o tom, jak se projekt mění.
Sledování práce na projektu pouze přes aplikaci Trello, kde každý viděl, kdo na čem zrovna pracuje a jaké úkoly jsou hotové, pouze pokud se dotyčný člen týmu sám na to podíval.	O průběhu práce na projektu má každý přehled, díky společnému naplánování.
Každý člen si soubory v adresáři třídil a pojmenovával dle svého. Déle trvalo dané soubory najít.	Sjednocené třídění a pojmenovávání souborů a znatelně lepší přehlednost v adresářích. Hlavně se názvy složek, souborů a skriptů nejeví neprofesionálně.
V seznamu nahraných verzí v Collaboration chybělo pojmenovávání verzí a změn. Nešlo jednoduše poznat co se v jaké nahrané verzi změnilo.	Povinný systematický popis u nově nahraných verzí. Díky tomu už jde jednoduše dohledat konkrétní změny i verzi projektu.
Programátoři si pojmenovávali třídy, proměnných, metod, atd. dle svého.	Pravidla pro sjednocené pojmenovávání v programování. O dost se zpřehlednil kód.

Tabulka 4 - Porovnání starého postupu vývoje s novou metodikou (Zdroj: autor)

3.6 Nástroje pro podporu vývoje a metodiky

V této kapitole budou popsány pouze nástroje, které byly a některé stále jsou používány ve vývoji a v nově zavedené metodice a se kterými vývojový tým přisel do styku.

Unity Collaboration

Obdoba způsobu verzování systému git. Slouží ke správě verzí projektu a také k tomu, aby všichni vývojáři, co pracují na projektu, mohli pracovat se stejnou kopií projektu. Služba Collaboration je nativně implementována v enginu Unity (software pro tvorbu aplikací a her), tudíž není třeba žádný program třetích stran. Oproti gitu si Collaboration soubory ukládá k sobě na Unity uložiště a velikost uložiště je omezeno na aktuální předplacené licenci.

Nástroje pro řízení projektu

Zprvu se projekt řídil, plánoval a sledoval v aplikaci Trello. Aplikace, která v sobě obsahuje sloupec a ve sloupcích se tvoří jednotlivé úkoly, které lze mezi sloupci přesouvat. Ukázka Trella je zobrazena na obrázku 7.

Ovšem Trello má jisté nevýhody. Prvním je, že při velkém počtu úkolu v jednom sloupci, se seznam úkoly stává naprosto nepřehledným. Druhá nevýhoda je, že nelze jednoduše třídit, seřazovat a přehledně pracovat s nějakým vlastním označením u úkolů, které by zobrazovalo obtížnost a prioritu úkolu. Třetí nevýhodou je to, že si nelze vyfiltrovat úkoly, které má člen přiřazené. To znamená, aby mu aplikace vypsala pouze úkoly, které má v aktuálním sprintu udělat ve svém vlastním seznamu a nemusel se prodírat cizími úkoly. Trello se neosvědčilo jako dobrý nástroj pro rozsáhlé projekty, které pracují s velkým backlogem.

Všechny tyto problémy řeší webová aplikace ClickUp, ale pouze v placené licenci, která ovšem oproti jiným podobným aplikacím jako je Asana nebo Monday je finančně výhodnější. Navíc obsahuje jak sloupcové zobrazení (podobně jako Trello), tak i řádkové, což znamená, že úkoly jsou přehledně zobrazeny pod sebou. Také je možnost vytvořit v úkolu jednotlivé podúkoly, což je užitečné, pokud je potřeba složitý úkol rozepsat do jednotlivých menších úkolů.

Obrázek 22 - Ukázka aplikace ClickUp úkolů v řádkovém zobrazení (Zdroj: autor)

Úložiště dat

Vývojový tým používal pro správu a ukládání dat sdílené uložiště počítačovou aplikaci Google Disk, kde se ukládaly veškeré materiály určené pro vývoj. Při zavádění nové metodiky se používal ke správě dat firemní NAS, ovšem z důvodu častých problémům s připojením NAS, se opět tým vrátil ke Google Disku.

4. Závěr

Hlavním výstupem této práce je metodika, určená pro vývojový tým ve společnosti Sever Game Studio, kde práce není založená na práci jednotlivců, kteří jsou řízeni jedním projekt manažerem, ale na vzájemné spolupráci v týmu.

V první polovině praktické části je obsažena analýza vývoje a spolupráce členů na projektu, která má značné nedostatky, které jsou eliminovány v návrhu metodiky, dle sepsaných kritérii, které mají prospívat jak vývoji, tak i týmové práci.

V úvodní části byly definovány hlavní cíle práce, jimiž jsou analýza stávajícího postupu vývoje, návrh nové metodiky, její zavedení s hodnocením, pravidla pro programátory a grafiky v týmové spolupráci. Splnění všech těchto cílů je obsaženo v této práci, které jsou rozděleny, právě i za účelem přehlednosti, do dvou hlavních kapitol. První kapitola se zajímá o starý postup vývoje a druhá o nový postup, kde je navrhnuta nová metodika. V praxi se nejednalo o přímé sepsání návrhu na novou metodiku a ihned její úspěšné zavedení, ale byl to téměř zdlouhavý jednoroční proces zavádění a neustálého vylepšování postupu vývoje a spolupráce do podoby jaká je sepsána v této práci.

Metodika přinesla významně lepší spolupráci členů v týmu, co se týče komunikace a vzájemného zapojení do plánování vývoje, třídění a přehlednost souborů jak ve sdíleném adresáři, tak i přímo v enginu Unity, což přineslo rychlejší dohledatelnost souborů a práci s nimi. V podstatě díky jednomu sepsanému dokumentu s těmito pravidly se dosáhlo toto zlepšení.

Také se výrazně zpřehlednily kódy. Zefektivnilo se plánování vzhledem k časové náročnosti díky tomu, že se u každého úkolu určila priorita a kolik času zabere jeho splnění. Při starém postupu trvalo udělat hratelné demo jedné hry téměř rok. Při nové metodice vývoj hratelného dema nové hry zabral dva měsíce, jelikož díky určování těchto priorit a týmovému plánování, se pracuje na nejdůležitějších funkcionalitách pro hru.

Díky metodice také došlo ke zlepšení vztahů a celkového zapojení všech členů do vývoje díky pravidelným fyzickým týdenním schůzkám, kde se členové aktivně zapojovali do naplánování vývoje a mohli tím ovlivnit vývoj, což přineslo návrhy na nové funkcionality a nový obsah. Při starém postupu celou hru vymýšlel pouze projekt manažér, nyní do designu hry přispívá celý tým.

Ze SWOT analýzy se eliminovala na minimum hrozba ohledně špatně rozložené pracovní síly na projektu a i také demotivace z nekonečného vývoje díky společnému určení priority a časové náročnosti na úkolech. Ze slabých stránek se odstranila také špatná komunikace v týmu zavedením pravidelných fyzických schůzek.

Tato metodika se dá uplatnit pro jakýkoliv vývojový tým, který bude potřebovat splnit kritéria sepsané v kapitole 3.4.1, které zahrnují práci s týmem na dálku, viditelné výsledky, aktivní zapojení členů a zlepšení komunikace, přehled o celém stavu vývoje pro celý tým a práci na opravdu důležitých funkcionalitách.

Seznam použitých zdrojů

Agilní metodiky řízení vývoje software (Agile software development methodologies). In: *ManagementMania* [online]. Wilmington (DE) 2011-2020, 23.12.2016 [cit. 26.04.2020]. Dostupné z: https://managementmania.com/cs/agilni-metodiky-rizeni-vyvoje-software

BAIRD, Scott. 15 Video Games That Spent Way Too Long In Development. In: *Screenrant* [online]. 2.12.2016 [cit. 2020-05-03]. Dostupné z: https://screenrant.com/15-video-games-that-spent-way-too-long-in-development/

CHANDLER, H., 2013. The Game Production Handbook. 3rd ed. Burlington, MA: Jones & Bartlett Learning. ISBN-13: 978-1449688097.

KORKISHKO, Iryna. Top 6 software development methodologies. In: S*yndicode* [online]. 5.10.2017 [cit. 2020-04-27]. Dostupné z: https://syndicode.com/2017/10/05/top-6-software-development-methodologies/

MAKUCH, Eddie. Rockstar: More than 1,000 people made GTAV. In: *Gamespot* [online]. 7.10.2013 [cit. 2020-05-03]. Dostupné z: https://www.gamespot.com/articles/rockstar-more-than-1000-people-made-gtav/1100-6415330/

SHYLENOK, Pavel. Understanding the Roles of Game Dev Professionals. In: *Gamasutra* [online]. 15.1.2019 [cit. 2020-04-26]. Dostupné z: https://www.gamasutra.com/blogs/PavelShylenok/20190115/334322/Understanding_the_Roles_of_Game_Dev_Professionals.php

TOMÁNEK, Martin. *Řízení projektů agilního vývoje softwaru na základě PRINCE2 a Scrum*. Vysoká škola ekonomická v Praze, 2015. DOKTORSKÁ DISERTAČNÍ

PRÁCE. Fakulta informatiky a statistiky Katedra systémové analýzy. Vedoucí práce Doc. Ing. Zora Říhová, CSc.

TRČKA, Martin. Retrospektivní arkádová hra. Vysoké učení technické v Brně, 2010. Bakalářská práce. Fakulta informačních technologií Ústav počítačových systémů. Vedoucí práce Ing. Petr Pospíchal

TURKE, Zachary. Understanding the iOS Developer Program License Agreement. In: *Law of The level* [online]. 19.2.2015 [cit. 2020-04-26]. Dostupné z: https://www.lawofthelevel.com/2015/02/articles/licensing/bargaining-with-apple-understanding-the-ios-developer-program-license-agreement/

TWEEDIE, Steven. The Most Popular Game Of 2014 Was Made In Less Than Three Days. In: *Business Insider* [online]. 16.12.2014 [cit. 2020-05-03]. Dostupné z: https://www.businessinsider.com/flappy-bird-is-most-searched-for-game-according-to-google-2014-12

Seznam obrázků

Seznam tabulek

Pojmy a zkratky

Herní engine = softwar, který obsahuje obecné funkce používané v počítačových hrách, které zrychlují vývoj her.

Unity = herní engine

Blender = software pro tvorbu 3D grafiky

Trello, ClickUp = software pro řízení projektu

Hierarchy = okno v enginu Unity, které zobrazuje objekty ve scéně

Textura = reprezentuje povrch 3D modelu pomocí vektorové mapy.

Difúzní, albedo, normálová, height textura = odlišné typy textur pro reprezentaci povrchu

Materiál = mapování textury, umožňují simulovat vizualizaci 3D objektu

Backlog = nashromážděná nedokončená práce.

Bug = chyba v kódu

Iterace = opakovaná činnost

AAA hry = vysoko rozpočtové hry

VR = virtuální realita

AR = rozšířená realita

NAS = datové uložiště připojené k síti