
# Fusion Tower Defense

Daniel Malík V3A
David Dvořák V3A
Daniel Hlůšek V3A
Tomáš Hájek V3A

# Rozdělení týmu

Náš tým se skládal ze tří programátorů a jednoho grafika. Kvůli tomuto rozložení jsme se rozhodli zaměřit se spíše na funkční stránku hry, protože jsme nechtěli našeho grafikáře kompletně zničit neustálým malováním.

### Manager a programátor David Dvořák

Pracoval na implementaci samotného systému věží od vývinu systému věží až po vytvoření jednotlivých druhů. Taktéž implementoval systém na kombinaci věží a jejich vylepšování.

### Programátor Daniel Hlůšek

Měl přiřazen systém map od vytvoření samotného systému až po poskládání map dohromady do hratelné podoby. Následně měl na starost systém nepřátel, což se týká jak vytvoření systému na chování nepřátel, tvoření nových nepřátel tak vytvořil i samotné nepřátele nacházející se ve hře. V poslední řadě taktéž vytvořil levely a přiřadil vlny.

### Programátor Daniel Malík

Měl přiřazenou implementaci grafického rozhraní naší hry, což znamená implementace balíčku karet, main menu, pause menu a obchodu s vylepšeními. Dále pracoval na správě dat ve hře jako jsou peníze, vylepšení, eventy, atd a na implementaci systému spawnování nepřátel ve vlnách.

### Grafikář Tomáš Hájek

Měl za úkol samotnou tvorbu grafiky pro hru ať už kreslení nepřátel či UI. Taktéž se věnoval designu hry jako je logo, zasazení či prostředí. Zároveň byl využit pro testování hratelnost nových map.

# Game loop

Naše hra patří do žánru Tower Defense. Hry v tomto žánru jsou velmi jednoduché, hráč načte mapu, má povoleno stavět věže za herní měnu. Věže může vylepšovat nebo prodávat a každá věž funguje unikátním způsobem, ale jejich hlavní účel je útočit na nepřátelé. Ti se objevují ve vlnách a hráč má jednoduchý úkol, dokončit všechny vlny bez toho, aby nechal nepřátelé projít na konec mapy.

Takových her však existuje mnoho, takže jsme se pokoušeli vymyslet nějaký unikátní způsob, jak tento žánr oživit bez toho, aby to byla jen další Tower Defense hra. Nakonec nás napadlo několik způsobů.

## Kartičky místo výběru věží

První z nich je to, že místo stavění věží přímo musí hráč brát kartičky z balíčku karet. Díky tomu se do hry přidá další úroveň strategie, protože hráč musí tvořit strategii podle věží, které zrovna dostane.

## Spojování věží

Druhá unikátní vlastnost naší hry je to, že věže jdou stavět na sebe. Každá věž tak působí zároveň něco co jde postavit, aby hráč měl více ochráněného místa, ale také jako vylepšení pro jinou věž, pokud například nestíhá.

## Každá hra je unikátní

Posledním způsob byl inspirován jiným žánrem, Roguelike. V tomto žánru je každé zapnutí hry kompletně unikátní, většinou tvorbou nové mapy, a zároveň nejdůležitější faktor tohoto žánru je permanentní smrt. Pokud hráč prohraje, musí začít kompletně od začátku, s novou mapou a bez všech vylepšení co si nasbíral. V naší hře implementujeme všechny tyto systémy. Hráč si vybírá svoji cestu na náhodně vygenerované mapě, může si každou hru kupovat vylepšení, které však v dalším zapnutí ztratí, a pokud umře, musí začít od začátku.

# Co nám týmová práce přinesla

Tato týmová práce byla pro všechny z nás první velký projekt, kde jsme byli nuceni spolupracovat. Znamenalo to, že jsme se naučili komunikovat, plánovat práci pro ostatní, určovat si časové období, kdy nějaká věc musí být dodělaná a mnoho dalších soft skills.

## Git Version Control System

Velká část přínosů byla v rámci používání programu Git. Nikdo z nás zatím nepracoval v projektu s několika lidmi, takže nebyl nucen používat věci jako rozdělení projektu na branche a poté opětovné spojení na master branch. Díky tomu jsme se také naučili pracovat s merge konflikty, protože se často stalo, že několik z nás potřebovali upravovat stejný kód zároveň.

## Godot + C#

Další věc co nikdo z nás nedělal byl game engine Godot. Někteří z nás v minulosti pracovali v Unity, ale kvůli změně smluvních podmínek od Unity a zároveň jako šance se naučit něco nového si náš tým vybral jako game engine Godot. S tím jsme si také museli vybrat programovací jazyk, kterým po hlasování byl vybrán na plné čáře C# místo GDScriptu.

Tímto projektem jsme se tedy naučili nejen nový game engine, ale také používat nový programovací jazyk, protože ze školy známe jen Python a Javu.

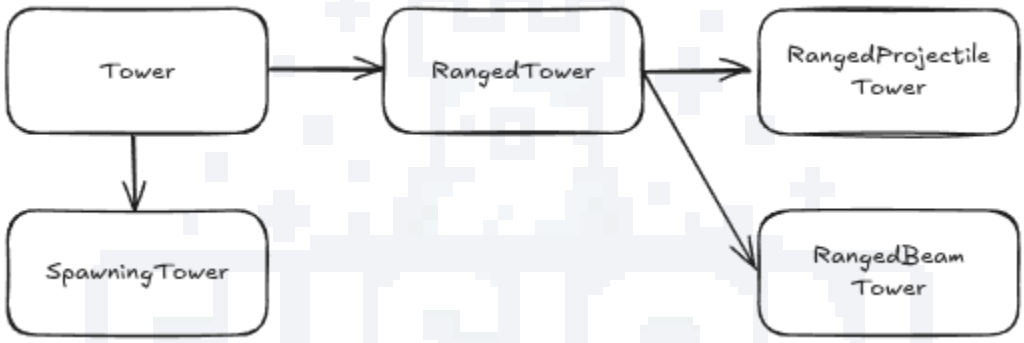
# Dokumentace členů týmu

Za touto stranou budou postupně vypsané části sepsané jednotlivými členy našeho týmu. Každý z nich popisuje část co v projektu udělal a na konci obsahuje hodnocení ostatních členů týmu dohromady s navrhovanou známkou.

# David Dvořák

Tower Defense hra by nebyla Tower Defense bez věží. Díky použití jazyku C# jsem měl v arsenálu mnoho užitečných věcí ze strany OOP jako například inheritance a polymorphism.

Díky OOP jsem si mohl věže poskládat jako hierarchii, kde začnu od nejzákladnějších vlastností ve třídě **Tower** a postupně stavím více specializované třídy až například po **RangedProjectileTower**.




**Hierarchie tříd pro věže**

Stejnou hierarchii jsem mohl udělat pro statistiky jednotlivých věží. Nejzákladnější informace dát do třídy **TowerInfo** a postupně skládat specializované třídy informací jako **RangedProjectileTowerInfo**.

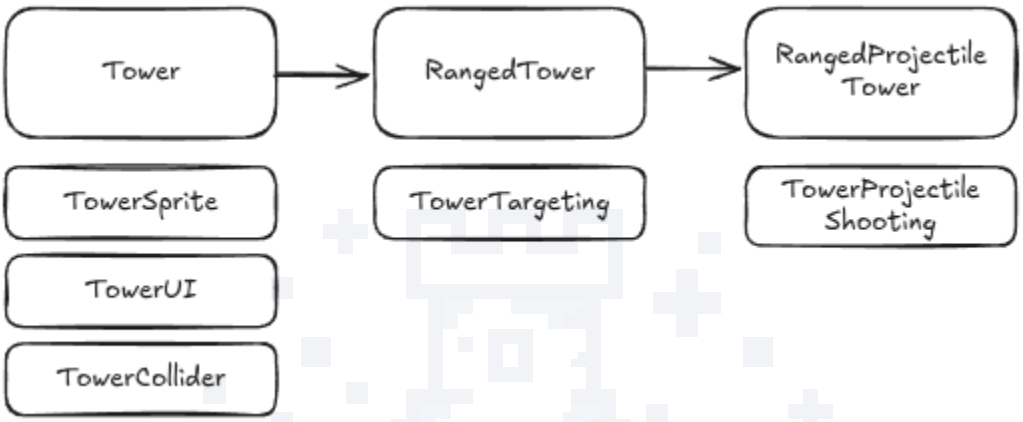
Třídy obsahují **virtual** metody, které si mohou přepisovat vyšší třídy na svou implementaci pokud to potřebují.

## Komponenty věží

Díky použití game enginu **Godot** jsem měl také možnost věž skládat z menších částí místo jednoho velkého bloku. Tomuto způsobu programování se říká kompozice a má **mnoho výhod**. Rozdělil jsem si tak jednotlivé funkce věží do menších podtříd, které vždy mají i v **Godotu** udělanou scénu obsahující všechny potřebné součásti pro daný komponent. Hlavní třída si pak jen načte jednotlivé komponenty, aby mezi sebou mohli komunikovat. **Godot** dovoluje **inheritovat** jednotlivé scény, takže jsem udělal scénu pro Tower, která obsahovala všechny

potřebné **komponenty**, tu pak inheritoval do scény **RangedTower**, která si zase vytvořila její potřebné komponenty, a takto pokračoval i pro zbytek scén.

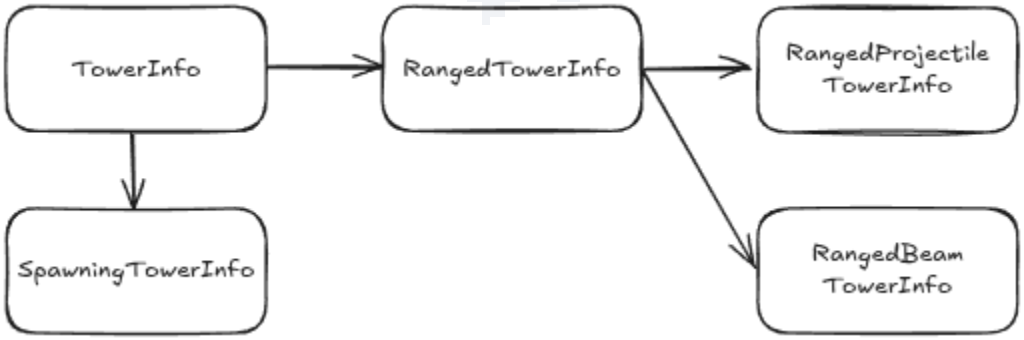
Díky tomuto systému jsem spojil sílu **OOP** jazyka se sílou **kompozice** a využil silné stránku obou způsobů implementace společné logiky.




**Hierarchie vytvořené scény**

## Vylepšování věží

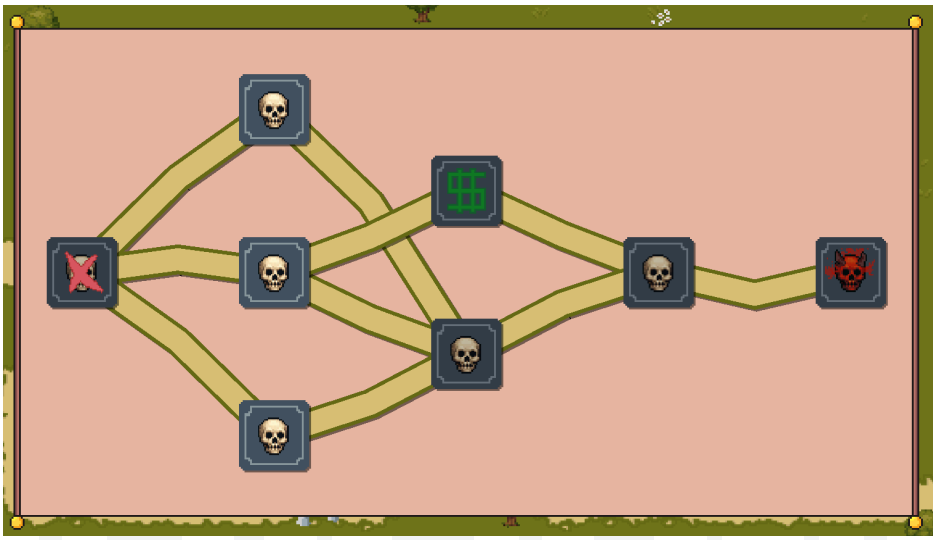
Poslední velkou částí našich věží je jejich vylepšování mezi sebou. To jsem implementoval přes speciální typ třídy **Resource** z **Godotu**. **Resource** udržuje data a **Godot** ji dokáže využít přímo v inspektoru pro editaci. Pomocí **[Export]** anotace jsem u každé věže vytvořil jednoduchý systém modifikace statistik dané věže, od **síly útoku** až po vylepšení **aplikované** na **další věž**, pokud se daná věž **položí na jinou**. Stejně jako u věží jsem zde využil inheritance.




**Hierarchie vlastností tříd**

## Výběr map

Pro unikátní zážitek z každé hry jsem přidal náhodně vygenerovanou mapu, kde si hráč může vybrat svou vlastní cestu podle toho, jak moc dobře hraje, protože obsahuje různé úrovní obtížností. Cesty jsou pěkně propojené a zároveň jde chodit jen na mapy kam vede cesta z mapy kde se hráč právě nachází.



## Hodnocení ostatních členů

S tímto týmem se mi pracovalo velmi dobře a pevně doufám, že jim se mnou také. Byli ochotní, rádi se učili nové věci a přežívali moje ne tak dobré manažerské dovednosti, i přes moji snahu.

**Daniel Malík** mnohokrát našel chyby v mém kódu, na které bych ani nepřišel. Zároveň velmi často prostě zmizel, takže někdy ho najít bylo doslova nemožné. I přes to bych mu dal jako **známku 1**, protože udělal velký kus práce.

**Daniel Hlůšek** byl také velmi dobrý člen týmu. Za velmi krátkou dobu dokázal prototypovat systém nepřátel a mnoho map, a i přes jeho neznalost Gitu se velmi rychle rozkoukal. Jeho názory byly občas trochu kritické, ale vždy se nám povedlo nějak domluvit, takže by měl také dostat **známku 1**.

**Tomáš Hájek** si z nás asi **známku 1** zaslouží nejvíc. Pracoval nepřetržitě, protože náš tým zrovna moc malovat nedokáže. Neustále jsem ho otravoval o další obrázky a on vždy došel s překrásným assetem co se do hry hodil. Jediná věc na kterou si mohu stěžovat je to, že Tom velmi často prostě úplně zmizel. Moc mi to nevadilo, přecejen maloval assety pro celý tým, ale definitivně komunikace mohlo být lepší.

# Daniel Malík

## Game Data

Třída GameData,  která je statická a slouží pro ukládání dat dané hry. Po zapnutí nové hry se tyto data resetují. Tato třída obsahuje:

- Počet shardů
- Počet celkově zabitých nepřátel
- Dictionary všech vylepšení a na jaké jsou úrovni
- Základní hodnoty počtu karet při načtení úrovně a modifikátory vylepšení

## Level Data

Třída LevelData je spojená s daty, které jsou jedinečné pro každou úroveň, po které se vždy resetují do původních hodnot. Tato třída je statická a obsahuje:

- Počet životů, goldů a zabitých nepřátel v dané úrovni
- Poslední a aktuální vlnu
- Cenu Karty
- Eventy pro změnu:
    - Životů a Goldů
    - Zabití nepřátel
    - Aktuální a maximální vlny
    - Ceny karty

## MenuButton

Slouží jako hlavní tlačítko pro ostatní menu. Obsahuje:

- BaseTexture - textura tlačítka, když se s ním neinteraguje
- HoverTexture - textura, která se nastaví, když je na tlačítku myš.
- PressedTexture - textura, která se nastaví když se na tlačítko klikne
- FontSize - velikost písma na tlačítku
- TextLabel - text na tlačítku

## Main Menu

Jednoduchý design, který využívá tomovi assety a můj MenuButton.


## Pause Menu

Je to součástí našeho autoload, což znamená že je to v každé scéně. Slouží to pro přesunutí zpět do main menu. Také využívá můj MenuButton.

## UpgradeButton

Extenduje MenuButton a slouží k nákupu vylepšení do všech úrovní v shop menu. Obsahuje:

- UpgradeId - id vylepšení, které slouží k identifikaci v GameData.
- ModiferLabel - který ukazuje aktuální modifikátor daného vylepšení.
- PriceLabel - který zobrazuje aktuální cenu za vylepšení o jednu úroveň.
- ShardPrice - základní cenu za vylepšení.
- ModifierUpgrade - hodnota o kterou se každý úroveň vylepší modifikátor.

## Shop Menu

Využívá UpgradeButton, kde vytvářím každé tlačítko ručně a nastavují jeho hodnoty. Poté je potřeba přidat příslušné id do GameData.

## Karta

Základ naší hry. Udělal jsem je tak, že každá kartička obsahuje věž, kterou má položit a všechnu logiku pro animace posouvání a zamykání kartiček.

## Balíček Karet

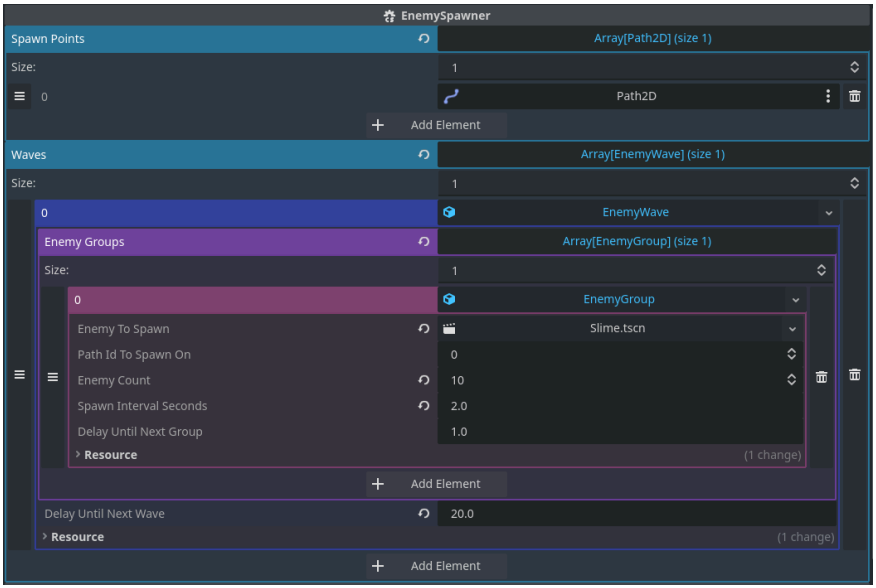
Obsahuje všechny kartičky. Dal jsem si s ním větší práci tak, aby animace a funkčnost kartiček. Obsahuje i tlačítko pro koupi další karty.


## Enemy Spawner

Kvůli potřebě jednoduchého systému pro dělání úrovní jsem vytvořil Node2D EnemySpawner scénu, která je na každé mapě. Tato scéna potřebuje array cest na mapě (Path2D) a array EnemyWave resource. Díky tomuto je celý proces vytváření vln na mapě velmi jednoduchý.

## EnemyWave resource

- Array EnemyGroup resource.



## EnemyGroup resource

- Scénu Enemy, kterou ma vytvořit.
- Počet kolikrát a sekundový rozestup mezi nimi.
- Pauzu mezi další EnemyGroup.

## Hodnocení ostatních členů týmu

### David Dvořák

Měl velký přínos do týmu. Jako vedoucí sice neexceloval, ale naprogramoval celý tower systém, který udělal modulární, aby bylo jednoduché v budoucnu přidávat spousty nových věží do hry. Také udělal rendering výběru map, který nebyl taky úplně jednoduchý. Za mě si david zaslouží pochvalu, protože jeho systém věží je velmi sofistikovaný a dobře strukturovaný.

### Tomáš Hájek

Díky Tomovi jsem mohl designovat UI. Dělal mi assety o, které jsem ho žádal často hodně rychle. Občas mu chyběla špetka kreativity, protože vždy když jsem ho o něco žádal bral to až moc doslovně co chci. Jeho assety mapy vypadají velmi pěkně a myslím si, že odvedl nádhernou práci s tím kolik toho musel namalovat.

### Daniel Hlůšek

Byl důležitou součástí týmu, i když ze začátku to vypadalo, že se mu nebude chtít pracovat v týmu a používat C# tak se nakonec přizpůsobil. Vytvořil nepřátele a všechny mapy ve hře až na tutorial mapu. Jeho mapy jsou velmi kvalitní a každá má něco do sebe. Rozdělil je i podle obtížností. Udělal i finální boss mapu, která je velmi náročná, ale i díky tomu zábavná.

# Daniel Hlůšek

## Nepřátelé

### C#

Esenčně se z programovacího hlediská nepřátelé skládají ze 3 scriptů. Script na animaci nepřátel, script na export hodnot nepřátel do Enginu a script samotných nepřátel. Script samotných nepřátel s nepřítelem pohybuje a ovláda co se stane v například v případě že nepřítel vezme poškození nebo zemře.

Tyto skripty se setkají ve scéně Enemy. Tato scéna tvoří šablonu pro nepřítele. Z této šablony se vytvoří scény pro samotné jednotlivé nepřítele, kde v

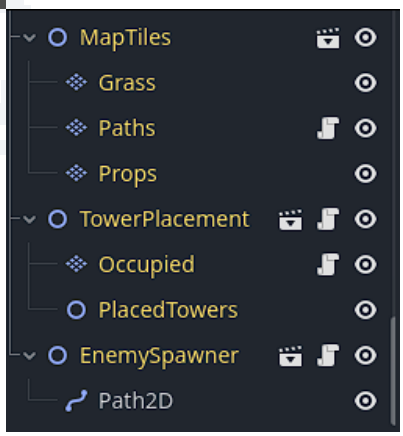
esenci všichni fungují stejně, ale zároveň dostatečně rozdílně aby pro hráče působily unikátně.

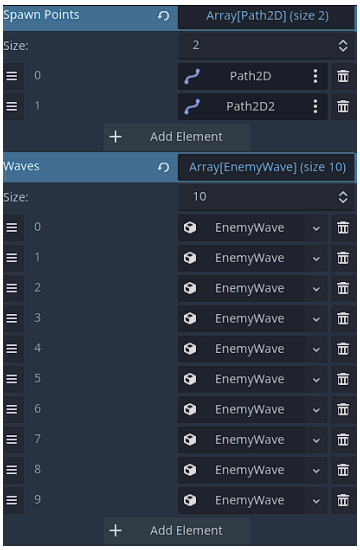
### Jednotlivý nepřátelé

1. Slime
    - Průměrně rychlý
    - Průměrně odolný
2. Fast slime
    - Nadměrně rychlý
    - Méně odolný
    - Větší peněžní odměna
3. Ultra fast slime
    - Extrémně rychlý
    - Menší velikost
    - Zemře na jednu ránu
4. Slow slime
    - Pomalý
    - Odolnější
    - Lehce zvednutá peněžní odměna
5. Ultra slow slime
    - Velice pomalý
    - Velice odolný
    - Velmi velká peněžní odměna

### Mapy

Mapový systém byl kompletně vytvořen v Enginu bez potřeby skriptování. Systém byl navrhnut a vytvořen tak aby bylo jednoduché, podobně jako u nepřátel, vytvořit scénu co dědí z předlohy pro mapy. Po vytvoření scény co která dědí z předlohy pro mapy se již dá velice jednoduše poskládat nová mapa v několika vrstvách




Tyto vrstvy reprezentují samotné prvky mapy, grass vrstva reprezentuje “podlahu” mapy. Paths reprezentuje cesty po kterých chodí nepřátelé. Props jsou stromy a kytičky na dekoraci mapy. Occupied určuje která políčka jsou zabraná a nedá se na ně stavět věže. EnemySpawner v sobě má jednotlivé cesty po kterých chodí nepřátelé.
Po vytvoření všech těchto vrstev je ještě potřeba vytvořit vlny ve kterých nepřátelé budou chodit. To se taktéž nastavuje v Enginu.

Tady se vloží cesty po kterých mohou nepřátelé chodit. Poté se vloží vlny, jednotlivé vlny se potom manuálně vytváří tím že se vloží skupiny po kterých chodí nepřátelé, u těchto skupin se nastaví co za nepřítele je v této skupině, kolik jich půjde a s jakým rozestupem a za jak dlouho půjde další skupina.

Takto bylo vytvořeno několik levelů, 3 easy, 3 medium, 3 hard a boss level. Rozdíl v těchto level je obtížnost v podobě množství nepřátel.

### Hodnocení týmu

**David Dvořák - známka 1** - Jakožto týmový manager se snažil přidělovat práci aby byla rozdělena rovnoměrně a nikdo nebyl přehlcen. Jakožto programátor byl taktéž pracovitý a spolehlivý a kdykoliv bylo potřeba cokoliv konzultovat tak byl cenným přínosem. Jakožto jediná výtka by se dalo zmínit nedostatek testování předtím než přidal svoji práci na GitHub.

**Daniel Malík - známka 1** - Tichí avšak velice pracovitý a šikovný programátor, který vždy v rozumném časovém rozmezí dodával všechnu svoji práci splněnou a funkční. Dala by se však vytknout komunikace, jelikož jsem si párkrát byl jistý že Dan byl vymazán z tohoto světa či při nejmenším utekl do jiné země.

**Tomáš Hájek - známka 1** - Vždy dodal grafiku načas a když byly potřeba změny tak neváhal a rychle upravil. Jakožto jediný grafik týmu měl mnoho práce za čež si myslím zaslouží ocenění.

# Tomáš Hájek

## Úvod

Naše týmová práce spočívala ve vývoji vlastní 2D tower defense hry s fantasy tématikou. Po společné diskusi jsme se rozhodli pro pixel artový styl, který je nejen vizuálně atraktivní, ale zároveň nám umožnil efektivní práci s omezeným časem. Hru jsme pojmenovali Fusion, protože její hlavní herní mechanikou je kombinování věží pomocí karet – právě podle toho vzniká i název, který odkazuje na "fúzi" různých herních prvků.

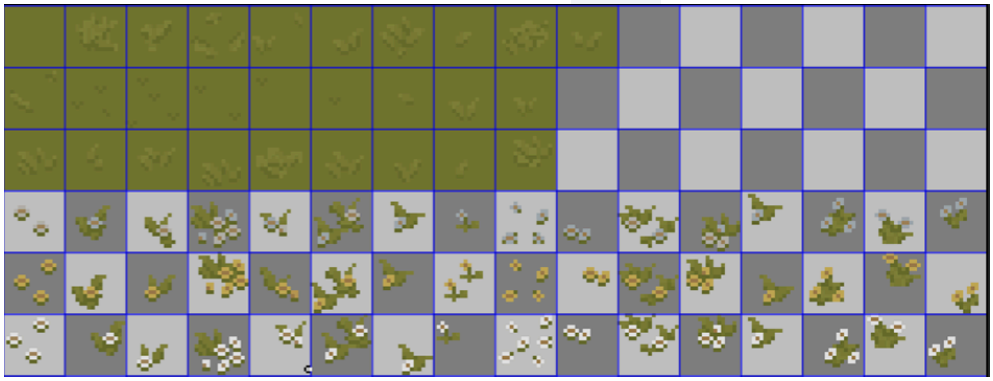
Ve hře hráč pomocí karet pokládá věže na mapu, aby čelil vlnám nepřátel. Každá věž má čtyři úrovně vylepšení a podle toho se mění její vzhled i síla.

V týmu jsem měl na starost veškerou grafiku – od návrhu dlaždic (tile setu), přes tvorbu jednotlivých věží a animací až po ikony, efekty nebo logo. Také jsem se částečně věnoval sound designu a vytvořil pár jednoduchých zvukových efektů.


Obrázek 1: Ukázka hry

## Gridový systém a tvorba mapy

Základem celé hry je gridový systém, který určuje, jak budou jednotlivé grafické prvky rozmístěny na mapě. Rozhodli jsme se pro grid o velikosti 16×16 pixelů, což znamená, že celý herní svět je složen z malých čtvercových dlaždic této velikosti. Tento přístup nám umožnil mít nad mapou úplnou kontrolu – všechny objekty do sebe přesně zapadají a hra díky tomu působí uspořádaně a přehledně.



Obrázek 2: Tileset pro trávu a květiny

## Vrstvení a skládání mapy

Mapa se skládá z několika vrstev, které na sebe postupně navazují:

1. Základní vrstva (terén) – vytváří základ prostředí.
2. Dekorativní vrstva – na základní terén se umisťují další asety jako cesty, kytky, stromy nebo keře. Tyto prvky dodávají mapě detail a atmosféru.
3. Herní vrstva – sem patří věže, pasti a další interaktivní objekty. I tyto prvky musí odpovídat velikosti gridu, aby správně seděly na mapu.

Obrázek 3: Tileset různých druhů cest a jejich variací

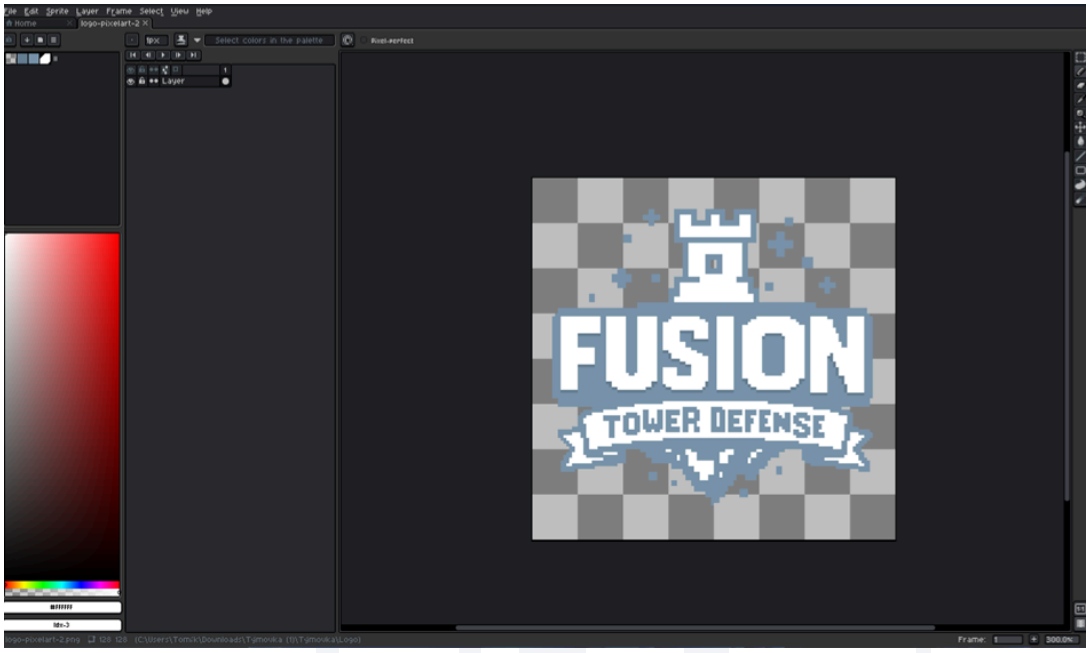
## Úprava cest a variace

Původně byly cesty navržené jako dlaždice 16×16 px, ale po prvních testech jsem se rozhodl je předělat na 32×32 px. Díky tomu působí detailněji a vizuálně výrazněji na herní mapě. Aby se cesty při opakovaném použití nezdály příliš monotónní, vytvořil jsem k hlavním variantám několik verzí, které se při skládání mapy střídají. Výsledkem je přirozenější vzhled bez opakujících se vzorů.

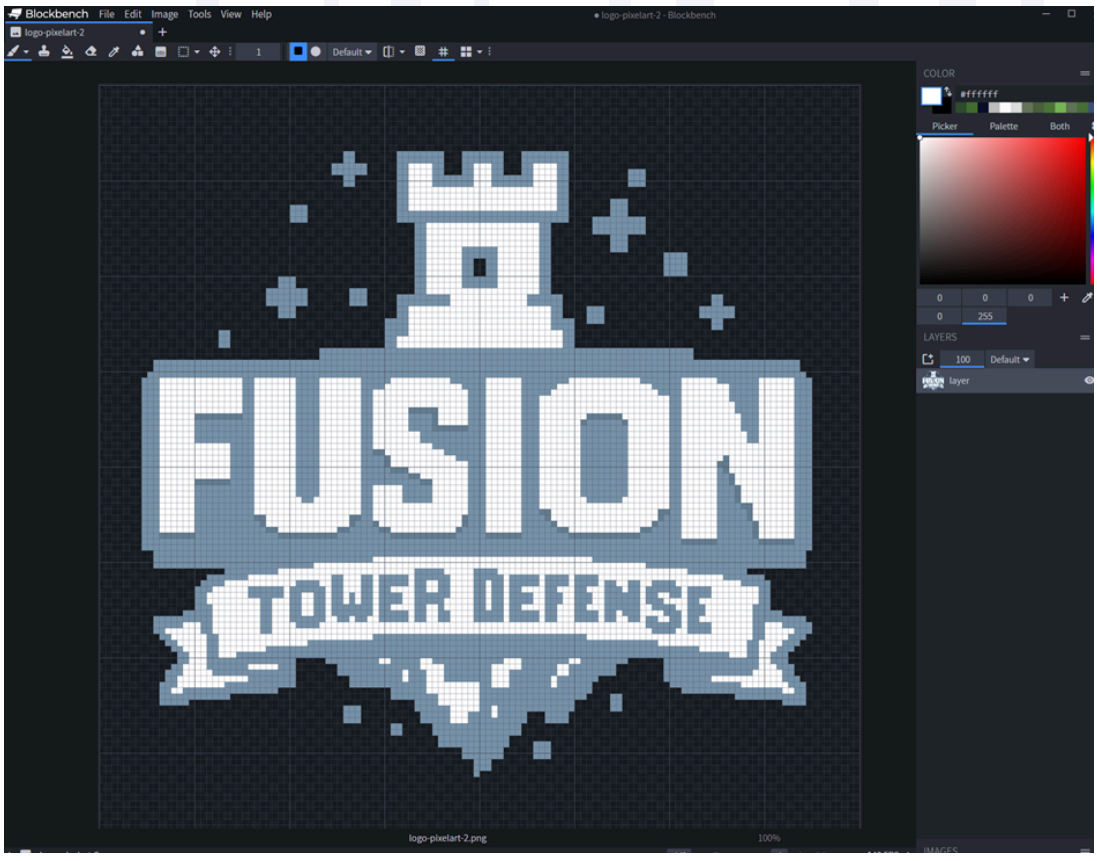
# Stylizace a nástroje pro tvorbu grafiky

### Výběr softwaru

Na začátku jsem zkoušel pracovat v programu Blockbench, který je určený hlavně pro 3D voxel/pixel grafiku. Rychle jsem ale narazil na jeho limity, pokud šlo o 2D tvorbu. Proto jsem přešel na program Aseprite, který se ukázal jako mnohem vhodnější.




Obrázek 5: Ukázka Aseprite




Obrázek 6: Ukázka Blockbench

V Aseprite se mi pracovalo výrazně lépe – prostředí je přizpůsobené pro pixel art, kreslení je plynulé, a hlavně umožňuje pohodlně vytvářet animace.

## Pravidla pro měřítko a velikost

Během práce jsem si ověřil, jak důležité je držet se jednotné pixelové velikosti. Na začátku jsem udělal chybu, kdy byly některé věže nebo textury větší nebo menší, než měly být. Například první věže jsem musel zmenšovat, aby se vešly do gridu.

## Věže

Všechny věže jsem dělal na gridu 32x64 px.

### Tiery

Každá věž má celkem **čtyři úrovně**. S každým vyšším tierem se zvyšuje její síla, rychlost útoku. Tento růst se odráží i v grafickém provedení.

- **Barracks** – spawnuje rytíře, kteří bojují s nepřáteli

Obrázek 7: Barracks - ukázka

- **Archer tower** – vystřeluje šípy na přicházející nepřátele, je navržená stejně jako kasárny, jen s rozdílnou střechou

Obrázek 8: Archer tower - ukázka

- **Mage** – používá magický paprsek, který zasahuje nepřítele.

Obrázek 9: Mage tower - ukázka

- **Trap Tower** – umisťuje pasti, které ubírají životy slima

Obrázek 10: Trap tower - ukázka

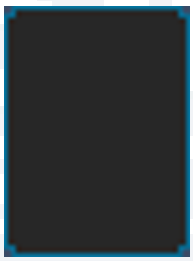
## Karty

Hráč nevybírá věže z menu, ale **hraje je pomocí karet**. Každá karta představuje jednu věž (tieru 1) a pokládá se na herní mapu. Abychom do hry přidali strategický rozměr, rozdělili jsme karty podle **vzácnosti**:

- **Common**

Obrázek 11: Common card

- **Rare**



Obrázek 12: Rare Card

- **Legendary**

Obrázek 13: Legendary Card

- **Mythical**

Obrázek 14: Mythical Card

## Nepřátelé

Hlavními a zatím jedinými protivníky ve hře jsou slimové, kteří fungují jako základní nepřátelská jednotka, je navržen tak aby se jednoduše dala měnit barva, pomocí které pak rozlišujeme vlastnosti jednotlivých slimů.

Tato jednoduchá změna barvy výrazně zvyšuje přehlednost pro hráče – i bez textu okamžitě pozná, s jakým typem protivníka má tu čest, a může podle toho reagovat.

Obrázek 15: Ukázka textury slima

## Logo hry

Vizuální identita hry začíná logem – je to první věc, kterou hráč uvidí.

### První návrhy a využití umělé inteligence

Přiznám se, že jsme ze začátku úplně nevěděli, jak by logo mělo vypadat. Abych si ujasnil směr, rozhodl jsme se využít umělou inteligenci, která mi pomohla vygenerovat několik prvních návrhů. Tyto návrhy mi poskytly inspiraci a pomohly určit celkový vizuální směr.

### Překreslení do pixel artu

AI návrhy ale nebyly použitelné přímo, protože neodpovídaly pixel art stylu zbytku hry. Proto jsem si jeden z nich vybral jako základ a překreslil ho do pixel artu. Tento proces byl poměrně časově náročný – bylo potřeba zachovat čitelnost, přehlednost i detail, ale zároveň neporušit estetiku malých rozměrů.


Obrázek 16: Výsledné logo jako pixel art

## Uživatelské rozhraní (UI)

Uživatelské rozhraní je klíčovým prvkem každé hry – umožňuje hráči snadno číst důležité informace a intuitivně ovládat hru.

### Ikony pro přehlednost

Pro zlepšení čitelnosti a orientace ve statistikách jsem vytvořil několik jednoduchých ikon.

Tyto ikony pomáhají hráči rychle pochopit, co se ve hře děje, bez nutnosti číst dlouhé popisky. Navíc jsou navržené v souladu s pixel-art estetikou celé hry.

Obrázek 17: Variace lebek jako ikonky (32x32 px)

Obrázek 18: Další ikonky (16x16 px)

## UI pozadí a tlačítka

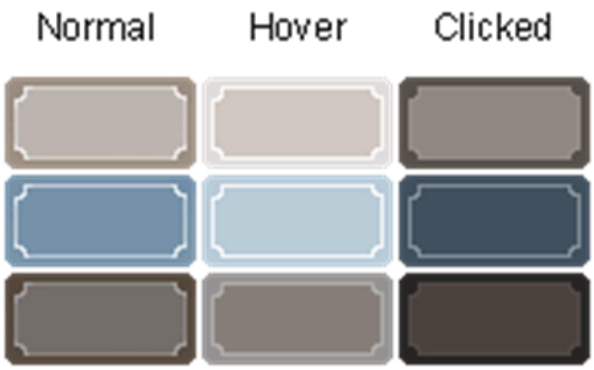
Kromě ikon jsem vytvořil i pozadí pro uživatelské rozhraní, které odděluje herní svět od ovládacích prvků. Je jednoduché, aby nerušilo, ale zároveň vizuálně zapadá do celkového stylu.

Navrhl jsem také tlačítka, která mají tři fáze stavu:

Normální (základní vzhled)

Hover (změna při najetí myší)

Clicked (změna po kliknutí)




Obrázek 19: Ukázka tlačítek

## Hodnocení ostatních členů týmu

Hodnocení je poměrně náročné protože jsem měl od zbytku týmu úplně rozdílnou práci tudíž jsem se moc nevměšoval do jejich části a nevěděl jsem co který programátor dělá. Většinou to pro mě fungovalo stylem, že mi někdo napsal že něco potřebuje a já to udělal.

### David Dvořák

David je hodně komunikativní, umí poradit, ale často byl trošku benevolentní co se týkalo organizace a hodně věcí se dělalo na poslední chvíli. Kdybych ho měl hodnotit známkou dal bych 1.

### Daniel Malík

Taky se s ním dělalo suprově hodně komunikativní, z mého pohledu, někdy když mi říkal co by potřeboval tak to znělo pro mě trošku zmateně. Taky bych ho hodnotil 1.

### Daniel Hlůšek

Byl trošku méně komunikativní, musel jsem se ho párkrát doptávat co potřebuje, ale zase mi hodně dobře dokázal podat co zrovna potřebuje, na dotazy odpovídal rychle a jasně hodnotil bych ho za 1.