Univerzita Hradec Králové
Fakulta informatiky a managementu
Katedra informatiky a kvantitativních metod

Aplikace teorie grafů: Počítačová hra

Diplomová práce

Autor: Bc. Schmidt Jakub
Studijní obor: Aplikovaná informatika

Vedoucí práce: RNDr. Ševčíková Andrea, Ph.D.

Hradec Králové Duben 2023

Prohlášení:

Prohlašuji, že jsem diplomovou práci zpracoval samostatně a s použitím uvedené literatury.

V Hradci Králové dne 24.4.2023 Jakub Schmidt

Poděkování:

Tímto bych chtěl poděkovat vedoucí mé diplomové práce RNDr. Andree Ševčíkové Ph.D. za metodické vedení a cenné rady týkající se této práce. Také bych chtěl poděkovat své manželce za podporu a pomoc s testováním.

Anotace

Tato práce se zaměřuje na vývoj a testování prototypu hry "Kingdom Defender" pro výuku teorie grafů. Hra je vytvořena v softwarových nástrojích Unity a Blender a implementuje algoritmy pro hledání cest, jako jsou prohledávání do šířky, Dijkstrův algoritmus a problém obchodního cestujícího. Práce poskytuje přehled dalších her, které obsahují prvky teorie grafů, a analyzuje jejich využití ve výuce. Uživatelské testování odhalilo pozitivní odezvu a užitečné návrhy pro zlepšení hry. Práce zdůrazňuje potenciál výuky prostřednictvím her a klade si za cíl inspirovat další výzkum v oblasti výuky matematiky a dalších předmětů. V budoucích krocích je vhodné zvážit rozšíření a zdokonalení hry, spolupráci s pedagogy a odborníky z oboru, a vytvoření vhodného výukového materiálu, který bude doplňovat herní zkušenost.

Annotation

Title: Applications of graph theory: Computer game

This thesis focuses on the development and testing of the prototype game "Kingdom Defender" for teaching graph theory. The game is created in software tools Unity and Blender and implements pathfinding algorithms such as breadth-first search, Dijkstra's algorithm, and the traveling salesman problem. The work provides an overview of other games that contain elements of graph theory and analyzes their use in teaching. User testing revealed positive feedback and useful suggestions for improving the game. The thesis emphasizes the potential of teaching through games and aims to inspire further research in the field of mathematics education and other subjects. In future steps, it is appropriate to consider expanding and refining the game, collaborating with educators and experts in the field, and creating suitable educational material to complement the gaming experience.

Obsah

Seznam obrázků

Seznam tabulek

Seznam ukázek kódu

1 Úvod

V dnešní době se stále častěji setkáváme s novými a inovativními způsoby výuky, které mají za cíl zvýšit zájem studentů o daný předmět a zlepšit jejich porozumění učiva. Jedním z takových přístupů je výuka prostřednictvím her, které dokáží udržet pozornost studentů a zároveň jim umožnit prakticky aplikovat teoretické znalosti. Tato práce se zabývá představením prototypu hry, jejíž základy vznikly jako hobby, ale postupně se z ní vyvinul nástroj pro podporu výuky teorie grafů.

Teorie grafů je matematický obor zabývající se studiem grafů, což jsou matematické struktury používané k modelování různých objektů a vztahů mezi nimi. Výuka teorie grafů je důležitá, protože se jedná o základní pojem v mnoha aplikacích a oborech, jako jsou například informatika, sociální vědy, biologie a mnoho dalších. Navzdory své důležitosti se však často setkáváme s obtížemi při výuce tohoto předmětu, a proto je zde potřeba najít efektivní a zábavné metody, které by usnadnily jeho pochopení.

Prototyp hry, který je předmětem této práce, byl vytvořen s využitím softwarových nástrojů Unity a Blender, což jsou běžně používané aplikace pro vývoj her a 3D modelování. Ve hře jsou při pohybu NPC využity algoritmy pro hledání cest, jako je prohledávání do šířky, Dijkstrův algoritmus a problém obchodního cestujícího.

Hra nabízí interaktivní prostředí, ve kterém si studenti mohou vyzkoušet různé algoritmy a učit se jejich principy prostřednictvím praktického řešení úkolů. Díky tomu se studenti mohou lépe zapojit do výuky daných algoritmů a získat hlubší porozumění pro teorii grafů a její aplikace.

Úvod práce pojednává o teorii grafů a algoritmech pro hledání cest. Následně se práce zaměřuje na použité technologie a seznámení se softwarem použitým pro vývoj aplikace. Poté je popsána vlastní implementace algoritmů a některé aspekty hry. V závěrečné části je uvedeno uživatelské testování, které zahrnuje vyhodnocení dotazníku od uživatelů. Tato část poskytuje cenné informace a umožňuje identifikovat možnosti pro další zlepšení aplikace.

V závěru práce jsou shrnuty dosažené výsledky a zkušenosti s vývojem a použitím prototypu hry v praxi, a jsou nastíněny další možné kroky směřující k vytvoření plně funkčního a komplexního výukového nástroje, který by mohl být široce používán ve výuce teorie grafů.

Tato práce tak představuje moderní přístup k výuce teorie grafů prostřednictvím her a ukazuje potenciál, který v sobě takové projekty mají, při snaze o zlepšení výuky a podpory studentů v jejich učení. Jejím cílem je inspirovat další výzkumy a projekty v oblasti výuky matematiky a dalších předmětů, a tím přispět ke zkvalitnění vzdělávání.

2 Cíl práce

Cílem této práce je vytvoření prototypu počítačové hry „Kingdom Defender“, ve které jsou implementovány některé metody hledání nejkratší či nejrychlejší cesty v grafech. Hra může sloužit jako podpora výuky teorie grafů a pro studenty poutavým způsobem, kdy své znalosti zdokonalují při hraní. V rámci diplomové práci byl zmapován názor studentů na využití této aplikace jako nástroje pro podporu studia.

3 Metodika zpracování

Vývoj hry, která má za cíl podporovat výuku teorie grafů, byl založen na metodologii získané z předchozích zkušeností a studia literatury. Algoritmy v jazyce C# byly implementovány na základě znalostí získaných z uvedených zdrojů a vlastních znalostí.

Hra byla vytvořena v herním enginu Unity, který byl zvolen z důvodu jeho širokého využití, flexibilitě a kompatibilitě s různými platformami. Pro modelování herních objektů byl použit program Blender, který je populární volbou pro 3D modelování, animace a tvorbu vizuálních efektů.

V rámci hry byly implementovány algoritmy pro hledání nejkratší cesty a trasy obchodního cestujícího. Tyto algoritmy byly zvoleny kvůli jejich významu v teorii grafů a jejich přenosu na řešení praktických problémů. Implementace algoritmů byla provedena v jazyce C#, který je běžně používán v Unity pro skriptování a programování herní logiky.

V konečné fázi vývoje hry bylo provedeno uživatelské testování pomocí nestandardizovaného anonymního dotazníku, který byl sestaven autorem pro účel ověřit funkčnost implementovaných algoritmů a vliv hry na zájem o předměty zabývající se teorií grafů, jakožto i celkovou hráčskou zkušenost. Obsahoval 5 otázek, viz Příloha B. Testování zahrnovalo skupinu studentů, kteří se seznámili s hrou a následně vyplnili dotazník hodnotící různé aspekty hry. Výsledky uživatelského testování byly pečlivě analyzovány a vyhodnoceny, což umožnilo autorovi identifikovat případné nedostatky a navrhnout možná zlepšení pro budoucí verze hry.

4 Teoretický rámec diplomové práce

V této části je vymezen pojem počítačová hra a je popsáno její využití ve vzdělávání. Podstatou teoretického rámce této práce jsou algoritmy pro hledání cest, konkrétně algoritmus prohledávání do šířky, Dijkstrův algoritmus a Problém obchodního cestujícího.

4.1 *Počítačová hra*

Počítačová hra je interaktivní zábavná forma digitálního média, ve které hráč ovládá jednoho či více postav nebo objektů s cílem splnit určité úkoly, dosáhnout cíle či porazit soupeře. Počítačové hry se vyznačují různými žánry, grafickým zpracováním a hratelností, a mohou být hrané na různých platformách, jako jsou stolní počítače, herní konzole nebo mobilní zařízení. [1]

Historie počítačových her sahá do 50. let 20. století, kdy byly vyvinuty první jednoduché hry na akademických počítačích. Jednou z prvních her byla "OXO" (1952) vytvořená Alexandrem S. Douglasem na počítači EDSAC. Průlomovým titulem byl "Pong" (1972) od společnosti Atari, který se stal první komerčně úspěšnou hrou a položil základ pro vývoj dalších herních titulů a platform. Od té doby se počítačové hry rozvinuly do mnoha žánrů a formátů, jako jsou adventury, strategie, simulace, sportovní hry, střílečky a další. [2, 3]

4.2 *Počítačové hry ve vzdělávání*

Využití počítačových her ve vzdělávání představuje nový a inovativní přístup k učení, který se stále více rozšiřuje. Hry mohou žáky motivovat a udržet jejich pozornost, díky čemuž si užívají učení více než při tradičních výukových metodách. Nicméně některé hry mohou být návykové, což může vést k nežádoucím účinkům na výkon studentů a jejich schopnost soustředit se na jiné činnosti. Výzkumy ukazují, že některé hry mohou pomoci ve vývoji sociálních dovedností, jako je komunikace, empatie a vyjednávání. Například hry s komunitními prvky, jako je Minecraft, podporují interakci mezi hráči, spolupráci a sdílení zdrojů a nápadů. Další hrou, která zlepšuje sociální dovednosti, je The Sims, kde se hráči musí starat o své virtuální postavy a zlepšovat jejich sociální interakce. [4]

Samozřejmě každá hra má jiný výukový potenciál, příkladem mohou být tyto hry zmíněné v [4, 5], které nabízejí jistý přínos pro studenty:

1. Portal: rozvoj logického myšlení, řešení problémů a kritického uvažování. Tato hra je založena na fyzice a prostorovém vnímání, což posiluje schopnost studentů analyzovat a řešit složité úkoly.
2. Minecraft: podpora kreativity, inženýrských dovedností a týmové práce. Hra umožňuje žákům stavět a objevovat ve virtuálním světě společně s ostatními, což zlepšuje sociální dovednosti a spolupráci mezi hráči.
3. Civilization: pochopení historických událostí, kultury a geopolitiky. Tato strategická hra zlepšuje analytické a rozhodovací dovednosti studentů, zatímco se učí o historii, náboženství, politice a vojenství.
4. Kerbal Space Program: aplikace fyzikálních zákonů a inženýrských dovedností. Hra umožňuje studentům navrhovat, stavět a řídit vesmírné lodě, což posiluje jejich znalosti fyziky, matematiky a inženýrských principů.

Dalším důležitým příkladem je polská hra This War of Mine. Jedná se o videohru od vývojářského studia 11 bit studios, která byla vydána v roce 2014. Hra se odehrává během války a místo toho, aby hráč hrál za vojáky, hraje za skupinu civilistů, kteří se snaží přežít v obléhaném městě. Hra klade důraz na morální rozhodování a ukazuje, jak mohou být etické otázky těžké a komplexní. [6]
Na rozdíl od výše zmíněných her se This War of Mine ve vzdělávání již používá. Polské střední školy se rozhodly začlenit hru do akademického kurikula pro starší studenty. Hra bude začleněna do výuky předmětů, jako jsou etika, filozofie, dějepis a další. Hra se nebude vyučovat ve třídách, pokud všem studentům není alespoň 18 let, protože obsahuje některá temná témata týkající se války a boje občanů o přežití. Rozhodnutí polských středních škol použít tuto hru ve vzdělávacím kontextu ukazuje dvě věci: za prvé, že povědomí o hodnotě videoher jako vyprávěcího média a legitimního způsobu učení se zdrojového materiálu rychle roste. Za druhé, že videohry jsou více než jen zábava pro děti. To dokazuje, že hratelnost a vyprávění "This War of Mine" jsou natolik cenné, že je lze prezentovat mladým lidem jako legitimní, akademický materiál ke studiu a analýze. Není zatím zcela jasné, jak bude hra začleněna do kurikula, ale "This War of Mine" rozhodně otevírá cestu

pro další skvělé hry, které by se mohly stát součástí školního vzdělávání jako legitimní díla umění a literatury. [7, 8]

4.2.1 Výhody používání počítačových her ve vzdělávání

Uvádí se několik klíčových výhod, které mohou počítačové hry přinést do vzdělávacího procesu. Jednou z těch hlavních je motivace. Hry mohou zvýšit zájem a motivaci studentů tím, že poskytují zábavný a interaktivní způsob učení. Tím se zvyšuje pravděpodobnost, že žáci se budou chtít dobrovolně učit a budou si nově získané znalosti lépe pamatovat. Další výhodou je podpora aktivního učení. Hry mohou žáky nutit přemýšlet, rozhodovat a řešit problémy, což zlepšuje kritické myšlení a rozvíjí dovednosti, které jsou důležité pro úspěch ve škole i v životě. Dalším aspektem, který mohou hry posílit je spolupráce. Některé hry vyžadují, aby žáci spolupracovali s ostatními, což může posílit sociální dovednosti a schopnost pracovat v týmu. Navíc hry umožňují individualizaci výuky. Mohou být přizpůsobeny potřebám a schopnostem jednotlivých žáků, což umožňuje diferenciaci výuky a podporuje osobní růst. [4]

Mezi další výhody zajisté lze zařadit zlepšení kognitivních schopností, jako je rozvoj logického myšlení, řešení problémů a rozhodování. Hry, jako je Portal, nabízejí jedinečné úkoly a problémy, které žáky nutí přemýšlet kreativně a strategicky. Rovněž hry mohou zlepšit spolupráci a týmovou práci, což je důležitá dovednost pro úspěch ve škole i v životě. [5]

4.2.2 Nevýhody používání počítačových her ve vzdělávání

Jednou z nevýhod využití her ve vzdělávání je, že mohou být časově náročné. Učitelé a rodiče musí vyčlenit čas na kontrolu hraní a zajištění, že se hry používají efektivně a s výukovým účelem. Za nevýhodu lze považovat náklady na pořízení potřebného softwaru a hardwaru, který může být pro některé školy a rodiny finančně náročný.

4.3 *Teorie grafů*

Základní pojmy z oblasti teorie grafů používané v diplomové práci by měly být pro čtenáře známé, lze je nalézt v [9]. V následujících kapitolách budou popsané algoritmy využívané ve hře.

4.3.1 Prohledávání do šířky

Prohledávání do šířky (*BFS - Breadth-First Search*) je jedním z nejjednodušších algoritmů pro vyhledávání v grafu a archetypem mnoha důležitých grafických algoritmů. Například Primův nebo Kruskalův algoritmus pro hledání minimální kostry grafu a Dijkstrův algoritmus používají podobné myšlenky jako prohledávání do šířky. Máme-li graf G = (V, E) a vyznačený počáteční vrchol „s“, prohledávání do šířky systematicky prozkoumává hrany grafu, aby „objevilo“ každý vrchol, který je dosažitelný z vrcholu s. Pro každý vrchol dosažitelný z S odpovídá nejkratší cesta mezi těmito vrcholy. Časová složitost algoritmu je O(|V|+|E|), kde |V| je počet vrcholů a |E| je počet hran grafu. Název algoritmu vychází ze způsobu procházení stromu grafu, kde se z počátečního bodu prozkoumají jeho sousedi a následně sousedi sousedů a tak dále, dokud nejsou prohledány všechny dosažitelné vrcholy nebo v případě hledání cesty mezi určitými vrcholy, dokud není dosažen cílový vrchol. [10]

```
function BFS(graph, start):
    // inicializace
    frontier = Queue()
    visited = List()
    frontier.enqueue(start)
    visited.add(start)

    // prohledávání
    while not frontier.isEmpty():
        current = frontier.dequeue()
        for neightbor in graph.neighbors(current):
            if neightbor not in visited:
                visited.add(neightbor)
                frontier.enqueue(neightbor)
```

Ukázka kódu 1 Pseudokód pro prohledávání do šířky **[11]**

V tomto pseudokódu se využívá fronta k ukládání vrcholů, které ještě nebyly prozkoumány, a visited k ukládání již navštívených vrcholů. Algoritmus začíná inicializací fronty („frontier") a přidáním počátečního vrcholu. Poté, dokud není fronta prázdná, se vyjme první vrchol z fronty, prozkoumají se jeho sousedé a pokud nebyli již navštíveni, přidají se do fronty a označí se jako navštívení. Tímto způsobem procházíme postupně všechny vrcholy, které jsou dosažitelné z počátečního vrcholu.

Důkaz správnosti BFS algoritmu popsaný v [10] je založen na několika klíčových pozorováních:

1. V každém kroku BFS je vzdálenost mezi zdrojovým vrcholem (start) a vrcholem odebíraným z fronty (current) rovna nejkratší možné vzdálenosti mezi těmito vrcholy v grafu. To je způsobeno tím, že BFS zkoumá vrcholy ve stejném pořadí, v jakém jsou objeveny při prohledávání grafu po vrstvách.
2. Jakmile je vrchol označen jako navštívený, už nebude znovu zařazen do fronty. To zajišťuje, že algoritmus nezkoumá stejný vrchol opakovaně a že se hodnota vzdálenosti pro daný vrchol nemění.
3. V průběhu provádění BFS je fronta Q řazena podle vzdáleností od zdrojového vrcholu. To znamená, že vrcholy s menší vzdáleností jsou zkoumány před vrcholy s větší vzdáleností. To zajišťuje, že algoritmus zkoumá vrcholy ve správném pořadí a správně určuje nejkratší vzdálenosti mezi zdrojovým vrcholem a ostatními vrcholy.
4. Když algoritmus BFS skončí, fronta je prázdná a všechny vrcholy, které lze dosáhnout ze zdrojového vrcholu, byly navštíveny. Výsledkem je, že vzdálenost mezi zdrojovým vrcholem a všemi ostatními vrcholy v grafu je správně určena.

Tyto body dohromady tvoří důkaz správnosti BFS algoritmu a zaručují, že algoritmus správně najde nejkratší vzdálenosti mezi zdrojovým vrcholem a všemi ostatními vrcholy v grafu.

4.3.2 Prohledávání do hloubky

Prohledávání do hloubky (*DFS – Depth-first search*), je, jak název napovídá, algoritmus prohledávající „hlouběji“ v grafu vždy, kdy když je to možné.

Prohledávání do hloubky prozkoumává hrany z nejnověji objeveného vrcholu, který má stále neprozkoumané hrany z něj vycházející. Jakmile jsou prozkoumány všechny hrany z tohoto vrcholu, prohledávání se vrací zpět a prozkoumává hrany z vrcholu, z něhož byl původní objevený vrchol nalezen. Tento proces pokračuje, dokud nejsou objeveny všechny vrcholy dosažitelné z původního počátečního vrcholu. Časová složitost algoritmu je O(|V|+|E|), kde |V| je počet vrcholů a |E| je počet hran grafu. [10]

```
function DFS(graph, start):
    visited = List()
    time = 0
    DFS_VISIT(graph, start, visited, time)

function DFS_VISIT(graph, current, visited, time):
    time += 1
    current.d = time
    visited.add(current)
    for neighbor in graph.neighbors(current):
        if neighbor not in visited:
            neighbor.pred = current
            DFS_VISIT(graph, neighbor, visited, time)
    time += 1
    current.f = time
```

Ukázka kódu 2 Pseudokód pro prohledávání do hloubky **[10]**

Průběh algoritmu v tomto pseudokódu vypadá následovně: na začátku je vytvořen seznam navštívených vrcholů a globální proměnná pro čas. Poté se spustí funkce „DFS_VISIT“ s vrcholem, od kterého chceme zahájit prohledávání. Funkce „DFS_VISIT“ provádí hlavní část algoritmu tím, že ukládá informace o objevení a dokončení zpracování každého navštíveného vrcholu pomocí časových razítek „d“ a „f“. Pro každý navštívený vrchol zvýší čas o 1, nastaví čas objevení a označí vrchol jako objevený, přičemž ho přidá do seznamu navštívených vrcholů. Následně prochází všechny sousední vrcholy aktuálního vrcholu. Pokud sousední vrchol není v seznamu navštívených vrcholů, nastaví jeho předchůdce na aktuální vrchol a rekurzivně zavolá funkci „DFS_VISIT“. Po zpracování všech sousedních vrcholů označí vrchol jako dokončený, zvýší čas o 1 a nastaví čas dokončení. Algoritmus DFS

takto efektivně prozkoumává graf a udržuje správné informace o vrcholech a jejich stavu. [10]

Důkaz správnosti popsán v [10] říká, že DFS je založen na vlastnostech intervalů časů objevení a navštívení vrcholů a na existenci objevených cest mezi vrcholy. V následujících bodech jsou popsány obě tyto vlastnosti, a jak souvisí s DFS.

1. Vlastnost intervalů časů: Pokud máme dva vrcholy „u“ a „v**“**, pak buď intervaly [u.d, u.f] a [v.d, v.f] jsou disjunktní, nebo jeden interval je zcela obsažený v druhém. To znamená, že v době, kdy je jeden z vrcholů objevený, je druhý buď neobjevený, nebo navštívený. Tato vlastnost je důležitá pro ujištění, že DFS správně prochází graf a že každý vrchol je navštíven právě jednou.
2. Vlastnost objevených cest: Vrchol „v“ je potomkem vrcholu „u“ v hloubkovém lese právě tehdy, když v době objevení vrcholu „u“ existuje cesta z **„**u“ do „v“ tvořená pouze objevenými vrcholy. Tato vlastnost ukazuje, že DFS efektivně objevuje nové vrcholy a zároveň zajišťuje správnou konstrukci hloubkového lesa.

Z těchto vlastností můžeme odvodit, že DFS správně prochází graf a konstruuje hloubkový les. V kombinaci s analýzou časové složitosti algoritmu, která byla zmíněna výše, lze dospět k závěru, že DFS je korektní a efektivní algoritmus pro prohledávání grafů. [10]

4.3.3 Dijkstrův algoritmus

Dijkstrův algoritmus je jedním z nejznámějších a nejpoužívanějších algoritmů pro řešení problému jednoho zdroje nejkratších cest v grafu s nezápornými hranami. Hlavní myšlenkou algoritmu je skenovat vrcholy v pořadí podle jejich rostoucích vzdáleností od zdrojového vrcholu. Pro každý vrchol se vypočítá jeho provizorní vzdálenost, která se aktualizuje během průběhu algoritmu. [11]

```
function Dijkstra(graph, start):
    // inicializace
    dist = Dictionary()
    prev = Dictionary()
    frontier = PriorityQueue()
    for v in graph.vertices:
        dist[v] = infinity
        prev[v] = null
    dist[start] = 0
    frontier.enqueue(start, 0)

    // prohledávání
    while not frontier.isEmpty():
        u, u_dist = frontier.dequeue()
        for v in graph.adjacent_vertices(u):
            alt = dist[u] + graph.edge_weight(u, v)
            if alt < dist[v]:
                dist[v] = alt
                prev[v] = u
                frontier.enqueue(v, dist[v])
```

Ukázka kódu 3 Pseudokód pro Dijkstrův algoritmus **[11]**

V pseudokódu se využívá prioritní fronta k ukládání vrcholů podle jejich vzdálenosti od počátečního vrcholu. Algoritmus začíná inicializací dist a prev pro každý vrchol, kde dist představuje aktuální nejkratší vzdálenost od počátečního vrcholu a prev ukládá předchozí vrchol v nejkratší cestě. Počáteční vrchol se nastaví na vzdálenost 0 a vloží se do prioritní fronty. Poté, dokud není fronta prázdná, se opakovaně vyjme vrchol s nejnižší vzdáleností z fronty, prozkoumá jeho sousedy a pokud je nalezena kratší cesta k sousedovi, aktualizuje se jeho vzdálenost a předchozí vrchol a soused se vloží zpět do fronty s novou prioritou. Tímto způsobem postupně najdeme nejkratší cestu ke všem vrcholům v grafu.

Důkaz správnosti Dijkstrova algoritmu je proveden ve dvou krocích:

1. Ukázat, že algoritmus navštíví (projde) všechny vrcholy dosažitelné z počátečního vrcholu „s“. Pokud existuje nějaký vrchol „v“, který je dosažitelný z s, ale není navštíven algoritmem, můžeme najít nejkratší cestu „p“ mezi „s“ a „v“. Když uvažujeme o této cestě, existuje vrchol „v_i“, který není navštíven před vrcholem „*v*“. Když je vrchol „v_i-1“ navštíven, jeho hodnota „d[v_i]“ je aktualizována na hodnotu menší než nekonečno. To znamená, že vrchol „v_i“ musí být nakonec navštíven, což je v rozporu s předpokladem, že v není navštíven. [11]
2. Ukázat, že skutečné a provizorní vzdálenosti se shodují, když je vrchol navštíven. Zvažte první okamžik, kdy je vrchol „v“ navštíven s hodnotou „μ[v] < d(v)“. Představme si nejkratší cestu „p“ mezi „s“ a „v“ a vrchol „v_i“, který není navštíven před vrcholem „v“. Když je vrchol „v_i-1“ navštíven, jeho hodnota „d[v_i]“ je aktualizována na hodnotu „μ(v_i)“. V okamžiku, kdy je vrchol v navštíven, hodnota „d[v_i]“ je menší než nebo rovna hodnotě „μ(v_k)“, což je menší než hodnota „d[v_k]“. To znamená, že vrchol „v_i“ by měl být navštíven místo vrcholu „v_k“, což je v rozporu s předpokladem. [11]

Tímto způsobem je Dijktrův algoritmus důkazem správnosti pro řešení problému jednoho zdroje nejkratších cest pro grafy s nezápornými hranami.

Celková časová složitost Dijkstrova algoritmu je závislá na použití prioritní fronty pro ukládání neskenovaných vrcholů s jejich provizorními vzdálenostmi jako klíči. Algoritmus používá operace „enqueue“ a „dequeue“ pro manipulaci s prioritní frontou a jeho časová složitost je

$$T_{Dijkstra} = O(m \cdot T_{enqueue}(n) + n \cdot (T_{dequeue}(n) + T_{enqueue}(n))),$$

kde „m“ je počet hran a „n“ je počet vrcholů v grafu. Tato složitost je založena na faktu, že každý vrchol je vkládán do fronty maximálně jednou a každá hrana je zpracována maximálně jednou. Závislost na „$T_{enqueue}$“ a „$T_{dequeue}$“ odráží dobu potřebnou k vložení vrcholu do fronty a jeho následné odebrání. [11]

V praxi je časová složitost Dijkstrova algoritmu často závislá na implementaci prioritní fronty. Při použití binární haldy, je časová složitost O(m * log n + n * log n) (vzhledem k tomu, že Tenqueue a Tdequeue mají obě složitost O(log n)). Existují však i jiné implementace prioritní fronty, které mohou vést k lepší časové složitosti,

jako například Fibonacciho haldy, které snižují časovou složitost na O(m + n * log n). [12]

4.3.4 Problém obchodního cestujícího

V problému obchodního cestujícího (*TSP – Traveling Salesman Problem*) je úkolem najít nejkratší možnou trasu, kterou může obchodník ujet při návštěvě n měst a následném návratu do výchozího bodu. Tento problém je úzce spojen s konceptem Hamiltonova cyklu, který je charakterizován okružní cestou, kde každé město, či vrchol grafu, je navštíven právě jednou. Problematika TSP může být modelována pomocí úplného grafu s n vrcholy, kde vrcholy představují města a hrany mezi nimi reprezentují cesty. Každá hrana má přiřazenou nezápornou váhu c(i, j), která odpovídá nákladům na cestu z města i do města j. Cílem obchodníka je najít Hamiltonovský cyklus s co nejmenšími celkovými náklady, které jsou dány součtem vah hran podél cesty. [10]

Problematika TSP má mnoho praktických aplikací, například v oblastech logistiky, plánování tras, nebo optimalizace výrobních procesů. Řešení TSP může mít zásadní dopad na efektivitu a nákladovou účinnost v těchto oblastech. Tento problém je známý svou obtížností a NP-úplností, což znamená, že není známo, zda existuje algoritmus, který by našel optimální řešení v polynomiálním čase. Vzhledem k významu a obtížnosti TSP bylo navrženo mnoho heuristických a aproximačních metod pro nalezení přibližných řešení s co nejmenší chybou. [13, 14]

- **NP-úplnost:** Určení, zda existuje řešení pro daný problém, může být z hlediska časové složitosti algoritmů velmi obtížné. Pokud neexistuje algoritmus s polynomiální časovou složitostí, který by pro daný problém našel řešení, označujeme tento problém za NP-úplný. NP-úplný problém je možné převést na každý jiný problém v třídě NP pomocí polynomiální redukce. Z toho vyplývá, že pokud bychom našli algoritmus, který by dokázal v polynomiálním čase řešit nějaký NP-úplný problém, bylo by možné ho použít pro řešení jakéhokoli problému z třídy NP, což by vedlo k tomu, že bychom mohli v polynomiálním čase řešit všechny NP-úplné problémy.

Zatím ovšem neexistuje důkaz, zda polynomiální algoritmus pro NP-úplné problémy existuje nebo neexistuje, a je to jedním z největších otevřených problémů v teoretické informatice. Do NP-úplných kombinatorických optimalizačních problémů patří například zmíněný problém obchodního cestujícího nebo barvení grafu. [15]

4.4 *Hry založené na teorii grafu*

Tato kapitola seznamuje čtenáře s hrami, které využívají principy teorie grafů v různých aspektech hry, jako je navigace, plánování a strategie.

4.4.1 Nine men's morris (Mlýn)

Mlýn je desková hra pro dva hráče, kteří se střídají v umisťování kamenů na hrací desku a posouvání těchto kamenů, aby vytvořili řadu tří kamenů ve stejném řádku nebo sloupci (tzv. „mlýn"). Pokud hráč vytvoří mlýn, má možnost odebrat jeden z kamenů soupeře z hrací desky. Hráč, který odebere všechny kameny soupeře, vyhrává hru. Hrací desku lze reprezentovat jako graf, kde vrcholy představují pozice na hrací desce a hrany představují možné tahy mezi těmito pozicemi. Tato reprezentace umožňuje snadno zkoumat možnosti tahů a plánovat strategii. Pomocí teorie grafů lze vyhodnotit různé pozice na hrací desce a určit, které tahy jsou nejvýhodnější, což může pomoci hráčům při rozhodování o nejlepším tahu v dané situaci. Teorie grafů také umožňuje zkoumat různé cesty v grafu, které mohou vést k vytvoření mlýna nebo k blokování soupeřova mlýna. Hráči mohou využít tuto analýzu k plánování své strategie a optimalizaci svých tahů. V průběhu hry se topologie grafu mění, když hráči přidávají a odebírají kameny. Teorie grafů může pomoci hráčům identifikovat taktiky, které využívají tyto změny ke svému prospěchu. Z tohoto pohledu lze hru analyzovat pomocí teorie grafů a zkoumat různé strategie a taktiky, které hráči mohou použít k dosažení vítězství. Teorie grafů tedy nabízí nástroje pro optimalizaci strategie a taktiky ve hře Mlýn, což může vést k lepšímu porozumění hry a zlepšení výkonu hráčů. [16]

4.4.2 Sudoku

Sudoku je hlavolam, který se skládá z mřížky 9x9 políček, které jsou dále rozděleny na 9 bloků o velikosti 3x3. Cílem je doplnit čísla od 1 do 9 do každého políčka tak, aby každé číslo v každém řádku, sloupci a bloku se vyskytovalo právě jednou. Z hlediska teorie grafů můžeme mřížku Sudoku reprezentovat jako graf, kde políčka mřížky představují vrcholy, a sousednost mezi políčky představuje hrany. Každé políčko je spojeno s jinými políčky v jeho řádku, sloupci a bloku. V průběhu řešení Sudoku se snažíme přidávat čísla do prázdných políček tak, aby platily omezení Sudoku - tj. aby žádné číslo nebylo v řádku, sloupci nebo bloku více než jednou. Nejdůležitější strategií pro řešení Sudoku je barvení grafu. Barvení grafu znamená přiřazení barvy (v případě Sudoku čísla) každému vrcholu tak, aby sousední vrcholy měly různé barvy. V Sudoku odpovídá každá barva číslu, které se snažíme doplnit do políčka. Pokud existuje řešení Sudoku, můžeme ho najít jako optimální řešení pro určitý typ barvení grafu.

4.4.3 Civilization

Civilization je série tahových strategických her, která byla poprvé vydána v roce 1991 a vytvořena Sidem Meierem. Hráči se ujímají role vůdce jedné z historických civilizací a snaží se ji rozvíjet a ovládnout svět prostřednictvím vědy, kultury, politiky, náboženství a vojenské síly. Cílem hry je obvykle vybudovat nejmocnější civilizaci a dosáhnout některého z vítězných podmínek, jako je například vědecké, kulturní nebo vojenské vítězství. [17]

Teorie grafů se v Civilization využívá k modelování různých aspektů hry, zejména v oblasti mapy světa a pohybu jednotek. Mapa ve hře Civilization může být považována za vážený orientovaný graf, kde vrcholy představují jednotlivá políčka na mapě a hrany představují možné cesty mezi nimi. Váhy hran mohou zahrnovat informace o terénu, jako jsou vzdálenosti, náročnost pohybu přes různé typy terénu nebo ovlivnění pohybu jednotek speciálními schopnostmi nebo technologiemi.

Teorie grafů se v Civilization také uplatňuje při hledání nejkratších cest mezi různými políčky na mapě, což je důležité pro efektivní plánování pohybu jednotek a logistiky. Algoritmy, jako je Dijkstrův algoritmus nebo A* algoritmus, mohou být použity k nalezení nejkratších cest mezi různými vrcholy v grafu.

V oblasti diplomacie a vztahů mezi civilizacemi se teorie grafů také uplatňuje při analýze sociálních sítí a vztahů mezi různými civilizacemi ve hře. To může pomoci hráčům identifikovat spojence, konkurenty a potenciální hrozby v rámci globálního uspořádání.

Obrázek 1 Hra Civilization VI [17]

4.4.4 Cities: Skylines

Cities: Skylines je městský simulátor a strategická hra od finského vývojářského studia Colossal Order a vydavatelství Paradox Interactive. Hra byla poprvé vydána v roce 2015 a hráči se ujímají role starosty, který má za úkol navrhovat a spravovat růst města. Hráči musí řešit různé problémy, jako je doprava, zásobování energií a vodou, vzdělávání, zdravotní péče, bezpečnost a životní prostředí, aby vytvořili prosperující a udržitelné město. [18]

Teorie grafů se v Cities: Skylines využívá především v oblasti dopravní infrastruktury a plánování dopravních tras. Městská dopravní síť může být považována za vážený orientovaný graf, kde vrcholy představují křižovatky a jiné body zájmu, jako jsou zastávky hromadné dopravy nebo výrobní a obchodní zóny, a hrany představují silnice, železnice, tramvajové tratě a další dopravní spojení. Váhy hran mohou zahrnovat informace o kapacitě cest, rychlostních limitech, typu dopravy a hustotě provozu. Teorie grafů se uplatňuje při hledání nejkratších a nejefektivnějších cest pro dopravu mezi různými částmi města.

Kromě dopravy a městského plánování může být teorie grafů v Cities: Skylines využita i pro analýzu a optimalizaci rozvodů elektřiny a vodního zásobování. Mřížky pro tyto sítě mohou být také modelovány jako grafy, kde vrcholy představují zdroje (elektrárny, čističky vody apod.) a spotřebiče (budovy, veřejná zařízení), zatímco hrany představují vedení a potrubí. Využitím teorie grafů lze optimalizovat distribuci těchto zdrojů a minimalizovat ztráty a náklady na infrastrukturu.

Obrázek 2 Hra Cities: Skylines [18]

4.4.5 Stellaris

Stellaris je 4X velká strategická hra (eXplore, eXpand, eXploit, eXterminate) od společnosti Paradox Interactive, která byla poprvé vydána v roce 2016. Hra se odehrává v reálném čase a hráči se ujímají role vedení mezihvězdného impéria, které se snaží zkoumat, kolonizovat a ovládnout galaxii. Hráči mohou provádět diplomatické akce, vyjednávat s jinými civilizacemi, provádět výzkum a technologický pokrok, řídit ekonomiku a zdroje a vést války. [19]

Teorie grafů se ve hře využívá pro modelování různých aspektů hry. Jedním z hlavních aspektů, kde teorie grafů hraje roli, je galaktická mapa, která představuje spojení mezi hvězdnými soustavami a cestami pro pohyb vesmírných lodí.

Galaktická mapa ve Stellaris může být považována za vážený orientovaný graf, kde jsou vrcholy interpretovány jako hvězdné soustavy a hrany jako hyperprostorové cesty mezi nimi. Váhy hran mohou být vnímány jako vzdálenosti mezi hvězdnými soustavami nebo čas potřebný k přepravě mezi nimi.

Teorie grafů se využívá také při hledání nejkratších cest mezi hvězdnými soustavami, což je důležité pro plánování efektivních tras pro lodě a řízení logistiky. Dále se využívá v oblasti diplomacie a vztahů mezi civilizacemi při analýze sociálních sítí a vztahů mezi různými impérii. To může pomoci hráčům identifikovat spojence, konkurenty a potenciální hrozby v rámci galaktického uspořádání.

Obrázek 3 Pohled na mapu galaxie ve hře Stellaris [19]

5 Použité technologie

V této části textu se jsou popsány technologie a nástroje, které byly využity při vývoji praktického projektu.

5.1 *Použité technologie*

V této části textu se jsou popsány technologie a nástroje, které byly využity při vývoji praktického projektu.

5.1.1 Blender

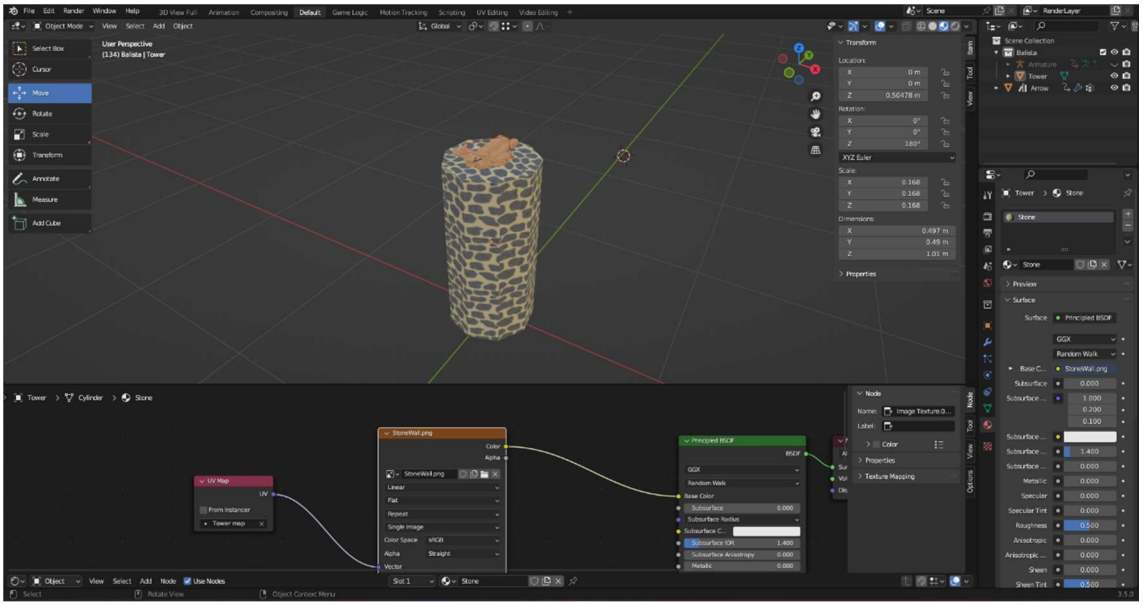
5.1.1.1 Úvod do Blenderu a jeho historie

Blender je komplexní, multiplatformní a open-source 3D software, který je vyvíjen neziskovou organizací Blender Foundation. Tento program se zaměřuje na 3D grafiku a animaci a poskytuje uživatelům široké spektrum nástrojů a funkcí pro tvorbu složitých a realistických scén a animací. Hlavním přínosem Blenderu je jeho bezplatná dostupnost, která umožňuje tvůrcům a umělcům, kteří si nemohou dovolit investovat do komerčního softwaru, využívat pokročilých funkcí pro tvorbu 3D grafiky a animací. Poprvé byl Blender uveden v roce 1995 jako interní nástroj pro NeoGeo a postupně získával na popularitě. V roce 2002 založil Ton Roosendaal neziskovou organizaci Blender Foundation s cílem zajistit nezávislý a otevřený vývoj Blenderu. Od té doby se Blender rozvíjel díky své aktivní a zapojené komunitě, která přispívá svými nápady, zpětnou vazbou a podporou. Díky tomuto přístupu se Blender stal konkurenceschopným nástrojem ve srovnání s komerčními 3D softwary, jako jsou Autodesk 3ds Max nebo Maya. V dnešní době je Blender používán profesionálními umělci, animátory, herními designéry, architekty a mnoha dalšími z celého světa. Jeho popularita a flexibilita z něj činí jednoho z nejvýznamnějších hráčů na poli 3D grafiky a animace. [20]

5.1.1.2 Základy a práce s Blenderem

Blender má poměrně komplexní uživatelské rozhraní, které se skládá z několika panelů, okének a záložek. Na začátku může být toto rozhraní pro nováčky složité, ale s časem se stává velmi efektivním a flexibilním. Navigace ve 3D prostoru

je základní dovednost, kterou je třeba zvládnout - lze ji provádět pomocí myši nebo klávesových zkratek. Základní manipulace s objekty zahrnují posun, rotaci a škálování, které lze provádět buď pomocí widgetů zobrazených přímo na objektu, nebo prostřednictvím klávesových zkratek.



Obrázek 4 Uživatelské rozhraní Blenderu (zdroj: autor)

5.1.1.3 Modelování v Blenderu

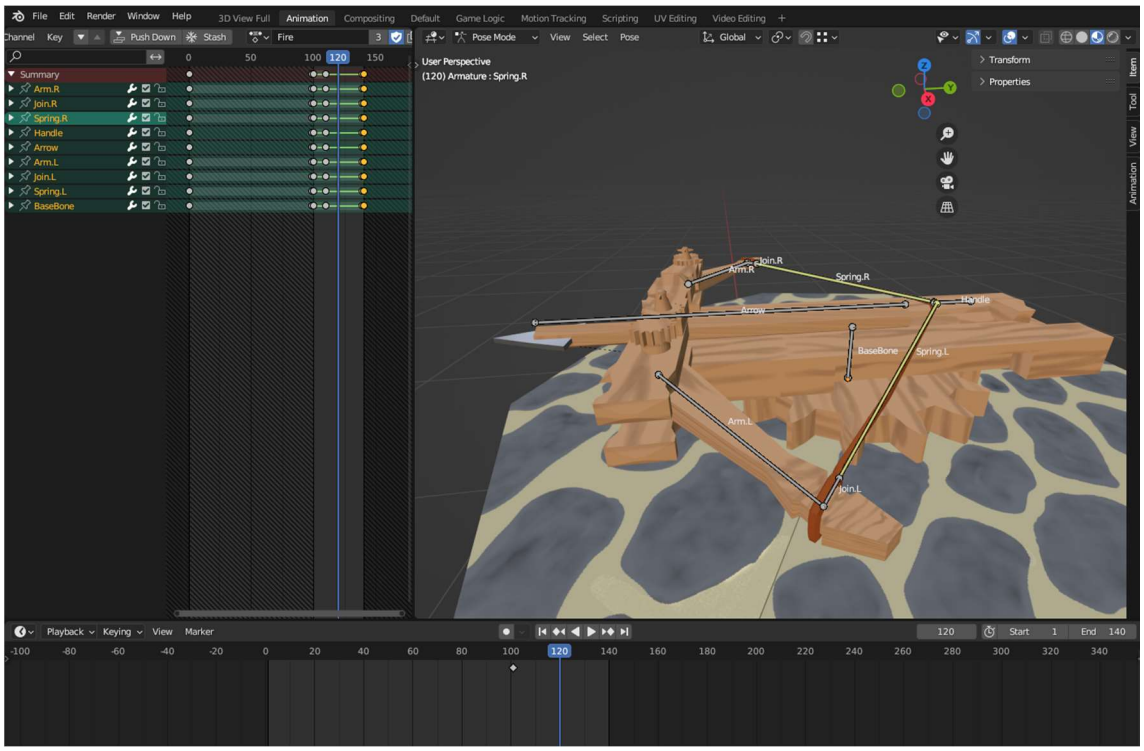
Modelování je klíčovou součástí práce s 3D grafikou, a Blender nabízí širokou škálu nástrojů pro vytváření a úpravu 3D modelů. Tyto nástroje zahrnují editaci mřížky a vertexů, práci s modifikátory, sculpting, retopologii a procedurální modelování. Každá z těchto technik má své vlastní specifické výhody a nevýhody, což umožňuje uživatelům zvolit si nejlepší přístup k vytváření modelů pro jejich konkrétní potřeby a účely.

5.1.1.4 Textury a materiály a jejich mapování

Textury a materiály jsou zásadní pro vytváření realistických a esteticky přitažlivých 3D scén. V Blenderu je možné vytvářet materiály a textury pomoc i Node Editoru, který umožňuje flexibilní a pokročilé úpravy materiálů. Pro správné zobrazení textur na modelech je nutné provést UV mapování, což je proces, při kterém jsou 2D textury správně aplikovány na 3D objekty. Proces zahrnuje "unwrap" objektu, což umožňuje přesné umístění textur na povrchu objektu.

5.1.1.5 **Animace a rigging**

Animace a rigging jsou klíčové součásti vytváření pohyblivých a interaktivních 3D objektů a postav. Rigging zahrnuje vytváření kostry, která řídí pohyb a deformace modelu. Tato kostra, známá také jako armatura, je složena z kostí, které se používají k ovládání různých částí objektu. Proces váh a skinning spočívá v přiřazení váh jednotlivým vertexům, které určují, jakým způsobem budou tyto vertexy ovlivněny pohybem kostí. Animace se provádí pomocí klíčových snímků (keyframes), které určují pozici, rotaci a škálování objektu v různých časech, a křivek, které určují, jak se objekt mezi těmito klíčovými snímky pohybuje.



Obrázek 5 Tvorba animací v Bledneru (zdroj: autor)

5.1.1.6 **Integrace Blenderu s Unity**

Blender a Unity jsou oba silné nástroje, které se často používají společně pro vytváření 3D her a aplikací. Aby bylo možné použít modely, animace a textury vytvořené v Blenderu v Unity, je nutné je nejprve exportovat do formátu, který Unity rozumí, jako je FBX. Poté je možné tyto soubory importovat do Unity a použít je

ve scénách a objektech. Při práci s oběma nástroji je důležité zachovat koordinátní systémy a jednotky, aby nedošlo k problémům s měřítkem nebo orientací.

5.1.1.7 Import a Export formátů a Unity

Blender podporuje řadu formátů pro import a export, což umožňuje snadnou integraci s ostatními nástroji a platformami, jako je například Unity. Pro práci s Unity je nejběžnějším formátem FBX, který podporuje modely, textury, materiály, animace a rigging. Dalšími podporovanými formáty jsou OBJ, COLLADA a glTF.

Dále je důležité si uvědomit, že při importu modelů do Unity může dojít k některým změnám v nastavení materiálů a osvětlení, proto je nezbytné provést úpravy přímo v Unity pro dosažení požadovaného vzhledu a výkonu. Optimalizace modelů a textur, například snižování počtu polygonů nebo použití komprimovaných textur, může být klíčové pro zajištění dobrého výkonu ve finálním projektu.

- **Formát FBX**: Formát FBX (**F**ilm**B**o**X**) je široce používaný a populární souborový formát pro 3D grafiku, který umožňuje přenos a výměnu dat mezi různými 3D softwarovými aplikacemi. Formát FBX byl původně vyvinut společností Kaydara, která byla později získána společností Autodesk. Formát je nyní podporován a udržován společností Autodesk, která je také známá svými produkty jako AutoCAD, 3ds Max nebo Maya. FBX podporuje širokou škálu datových typů, včetně modelů, materiálů, textur, animací, kosterního riggingu a dalších. Díky své univerzálnosti je FBX ideální pro výměnu dat mezi různými aplikacemi, jako jsou Blender, 3ds Max, Maya, Unity, Unreal Engine a další. [21, 22]

5.2 *Unity*

5.2.1 Co je to Unity

Unity je herní engine vyvíjený společností Unity Technologies umožňující vytváření her a aplikací. Poprvé bylo Unity představeno a vydáno v roce 2005 pro operační systém Mac OS X s cílem umožnit vývojářům vývoj her bez ohledu na jejich technické znalosti. Engine postupně rozšířil svou podporu pro různé desktopové, mobilní i konzolové platformy a v neposlední řadě také podporuje

vývoj pro virtuální realitu. Díky své dostupnosti a snadnému používání je oblíbený mezi začátečníky a také pro vývoj nezávislých her. V Unity lze vytvářet jak trojrozměrné, tak dvourozměrné hry nebo interaktivní simulace. Mimo jiné je engine využívaný také mimo herní průmysl a to třeba v automobilovém průmyslu, architektuře, strojírenství nebo ozbrojenými silami Spojených států. [23]

Unity disponuje poměrně jednoduchým uživatelským rozhraním a obsahuje mnoho nástrojů a funkcí, které usnadňují tvorbu her, jako je například editor scén a animací. Dále je vývojářům k dispozici Unity Asset Store, což je rozsáhlá knihovna předpřipravených assetů, díky kterým se může usnadnit a zkrátit potřebný čas k vytvoření hry.

Nicméně se lze setkat s vývojáři, kteří kritizují Unity, že má omezenou optimalizaci a v něm vytvořené hry mohou být náročnější na výkon, než hry vytvářené v jiných enginech. Dále má Unity určité licenční poplatky pro pokročilé funkce a pro možnost vydávání her pro herní konzole.

5.2.1.1 **Proč používat Unity**

Unity nabízí několik výhod pro vývojáře her a interaktivního obsahu. Některé z hlavních důvodů zahrnují:

- S Unity je možné snadno vytvářet hry a aplikace pro různé platformy bez nutnosti měnit kód.
- Editor Unity nabízí přehledné a přizpůsobitelné prostředí, které usnadňuje práci s objekty ve scéně, skripty a assety. To umožňuje rychlejší a efektivnější vývoj her.
- Unity má obrovskou komunitu vývojářů a bohatou zásobu zdrojů, jako jsou návody, dokumentace, fóra a Asset Store, což usnadňuje získání pomoci a zdrojů pro váš projekt.
- Díky integrovanému skriptovacímu jazyku C# a široké škále dostupných nástrojů a rozšíření můžete v Unity vytvářet prakticky jakýkoli druh hry nebo aplikace.
- Unity je optimalizováno pro různé platformy a zařízení, což zajišťuje dobrý výkon her a aplikací. K dispozici jsou také nástroje pro ladění

a optimalizaci, které pomohou dosáhnout co nejlepšího výkonu na cílových platformách.

5.2.1.2 Verze a licence Unity

Unity je k dispozici v několika verzích a licencích, které se liší podle rozsahu funkcí a podpory. V první řadě Unity nabízí zdarma verzi „Personal“, která je pro individuální vývojáře, malé týmy a nekomerční projekty. Poskytuje základní funkce Unity, avšak s omezenou úrovní podpory. Pro malé týmy a firmy s ročním obratem nižším než $200,000 je v nabídce verze „Plus“, která zahrnuje všechny funkce „Personal“ verze a navíc nabízí některé prémiové funkce a lepší podporu. „Unity Pro“ verze je určena pro středně velké a velké firmy s ročním obratem nad $200,000. Zahrnuje veškeré funkce Plus verze, pokročilé funkce a nástroje pro optimalizaci a neomezenou úroveň podpory. A jako poslední je verze Enterprise, která je určena pro velké organizace, které vyžadují individuální řešení a podporu.

5.2.2 Základní koncepty Unity

5.2.2.1 Hierarchie a scény

Scéna je prostor, ve kterém se nacházejí a interagují veškeré objekty ve hře nebo aplikaci. Hierarchie je zobrazení všech objektů ve scéně ve formě stromové struktury, která usnadňuje organizaci a navigaci mezi objekty. Objekty mohou být uspořádány do hierarchických vztahů, kde jeden objekt může být rodičem a jiný potomkem, což umožňuje sdílet vlastnosti a transformace mezi nimi.

5.2.2.2 GameObjecty a komponenty

GameObject je základní stavební jednotka ve scéně Unity, která může reprezentovat jakýkoli objekt, jako jsou postavy, budovy, světla nebo kamera. GameObject sám o sobě nemá žádné viditelné nebo fyzikální vlastnosti. Místo toho získává své chování a vzhled prostřednictvím komponent, které jsou přidány k němu. Komponenty mohou zahrnovat renderery, kolizní objekty, fyzikální

simulace, zvuky, skripty a mnoho dalšího. Díky komponentám lze GameObjecty snadno upravit a rozšiřovat.

5.2.2.3 **Assety a prefabrikáty**

Unity pojmem „asset“ označuje veškerý obsah používaný při tvorbě her. Může se jednat o FX (*special effects*) prvky jako jsou modely, textury, zvuky nebo animace, ale jsou jimi například i skripty a scény samotné. Lze je vytvářet samostatně a poté do projektu importovat a mohou být snadno použity na vícero místech ve hře. Například zvukový efekt může být použit pro více akcí ve hře. Prefabrikáty (*angl. Prefab*) jsou speciální druh assetů, které představují prefabrikované GameObjecty s přednastavenými komponentami a vlastnostmi. Prefabrikáty usnadňují opakované použití a sdílení objektů ve vašem projektu. Vytvořený prefabrikát, se může snadno umístit do scény, aniž by bylo třeba ručně konfigurovat všechny jeho komponenty. Pokud dojde k úpravě prefabrikátu, změny se automaticky projeví ve všech instancích tohoto prefabrikátu v projektu.

5.2.2.4 **Fyzika a kolize**

Unity obsahuje integrovaný fyzikální engine, který umožňuje simulovat realistické pohyby a interakce objektů. Pro práci s fyzikou ve scéně můžete přidávat komponenty, jako jsou Rigidbody (pro simulaci dynamiky objektů) a Collider (pro detekci kolizí a interakcí mezi objekty). Rigidbody určuje, jak se objekt pohybuje a reaguje na vnější síly, zatímco Collider definuje hranice a tvar objektu pro detekci kolizí. Tyto komponenty pracují společně, aby zajistily realistické chování objektů ve scéně, jako je například gravitace, tření, odrazy a interakce mezi objekty.

Fyzikální engine je vysoce konfigurovatelný a umožňuje nastavit parametry, jako jsou hmotnost, tření a odpor vzduchu pro každý objekt individuálně. Lze také vytvářet vlastní fyzikální materiály, které určují specifické vlastnosti interakce mezi objekty, jako je odrazivost nebo viskozita.

Kromě toho lze ve skriptech reagovat na události, jako jsou kolize a interakce mezi objekty. Toto například umožňuje spustit animaci nebo zvuk, když se dva

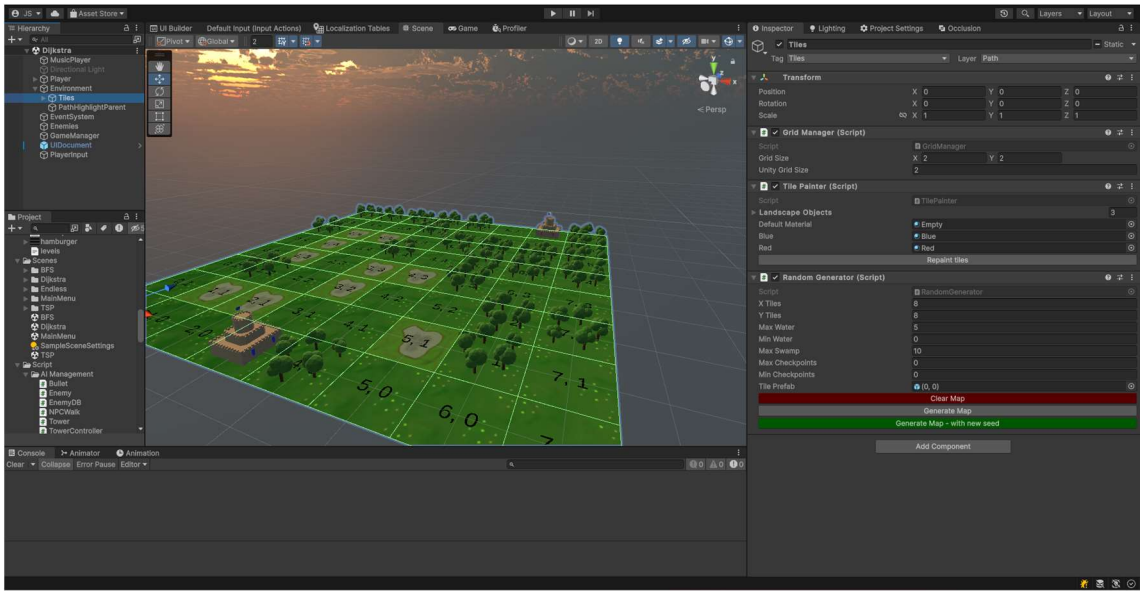
objekty střetnou, nebo změnit chování objektu v závislosti na interakcích s ostatními objekty.

5.2.3 Práce s editorem Unity

5.2.3.1 Uživatelské rozhraní editoru

Editor Unity je místo, kde vývojáři pracují na svých hrách a aplikacích. Uživatelské rozhraní editoru se skládá z několika panelů a oken, které usnadňují práci s objekty, komponentami, assety a kódem. Hlavní panely zahrnují:

- Scéna: Zobrazuje aktuální scénu, umožňuje manipulaci s objekty a umístění nových objektů do scény.
- Hierarchie: Zobrazuje stromovou strukturu objektů ve scéně, umožňuje navigaci a organizaci objektů.
- Inspektor: Poskytuje podrobné informace o vybraném objektu, včetně jeho komponent a jejich vlastností, a umožňuje úpravy.
- Projekt: Zobrazuje strukturu složek a assetů ve vašem projektu a umožňuje import, export a organizaci assetů.
- Konzole: Zobrazuje výstup skriptů, chybové zprávy a ladící informace.



Obrázek 6 Uživatelské rozhraní Unity (zdroj: autor)

5.2.3.2 Manipulace s objekty ve scéně

V editoru scény je možné vybírat, přesouvat, otáčet a škálovat objekty pomocí transformačních nástrojů. Tyto nástroje umožňují rychlé a přesné umístění a úpravu objektů ve scéně. Kromě toho umožňuje vytvářet, duplikovat a mazat objekty pomocí kontextových nabídek a klávesových zkratek.

5.2.3.3 Správce projektu a prohlížeč assetů

Správce projektu („Project window“) je součástí uživatelského rozhraní editoru Unity, který zobrazuje všechny assety a složky v projektu a umožňuje procházet, organizovat a spravovat assety. Správce projektu také poskytuje nástroje pro import, export, vyhledávání a filtrování assetů. V rámci správce projektu lze také tvořit a spravovat složky, které pomohou udržet projekt organizovaný.

Prohlížeč assetů („Asset Browser“) je integrován do správce projektu a zobrazuje náhledy a detaily o vybraném assetu, včetně jeho typu, velikosti, autora a dalších metadat. Prohlížeč assetů usnadňuje identifikaci a výběr vhodných assetů pro scénu.

Pro import assetů do projektu je možné použít standardní nabídku Import nebo přetáhnout soubory přímo do správce projektu z počítače. Unity automaticky rozpozná typ assetu a provede potřebné konverze a nastavení. Assety lze mimo jiné také získat z Unity Asset Store, který nabízí širokou škálu předem vytvořených zdrojů, jako jsou 3D modely, textury, zvuky, skripty a kompletní projekty, které můžou usnadnit a urychlit vývoj.

5.2.4 Skriptování v Unity

5.2.4.1 Jazyk C# v Unity

Unity používá jazyk C# jako primární skriptovací jazyk pro vývoj her a aplikací. C# je moderní, silně typovaný a objektově orientovaný jazyk, který poskytuje výkonné nástroje a funkce pro tvorbu interaktivních a komplexních scén. Díky jeho integraci s Mono a .NET frameworky, C# nabízí širokou škálu knihoven a API pro práci s grafikou, zvukem, fyzikou, sítěmi a dalšími.

5.2.4.2 Struktura skriptů a životní cyklus

Skripty v Unity mají základní strukturu třídy, která dědí od třídy „MonoBehaviour“. Třída „MonoBehaviour“ je základní třída pro všechny skripty Unity a poskytuje metody a události pro práci se životním cyklem objektů. Mezi tyto metody patří například „Awake“, „Start“, „Update“, „FixedUpdate“ a „OnDestroy“. Tyto metody se automaticky volají během různých fází životního cyklu objektu a umožňují inicializovat proměnné, provádět průběžné aktualizace a uvolnit zdroje při zničení objektu. [23]

```
public class Example : MonoBehaviour {
    // Voláno, když je skript načten
    private void Awake() {
        Debug.Log("Awake: Skript byl načten");
    }

    // Voláno před prvním snímkem Update
    private void Start() {
        Debug.Log("Start: Skript je připraven");
    }

    // Voláno každý snímek
    private void Update() {
        Debug.Log("Update: Každý snímek");
    }

    // Voláno každý snímek s pevnou frekvencí
    private void FixedUpdate() {
        Debug.Log("FixedUpdate: Každý snímek s pevnou frekvencí");
    }

    // Voláno před vykreslováním snímku
    private void LateUpdate() {
        Debug.Log("LateUpdate: Před vykreslováním snímku");
    }

    // Voláno při zničení objektu s tímto skriptem
    private void OnDestroy() {
        Debug.Log("OnDestroy: Objekt se skriptem byl zničen");
    }
}
```

Ukázka kódu 4 Ukázka životního cyklu skriptu (zdroj: autor)

5.2.4.3 . Přístup ke GameObjectům a komponentám

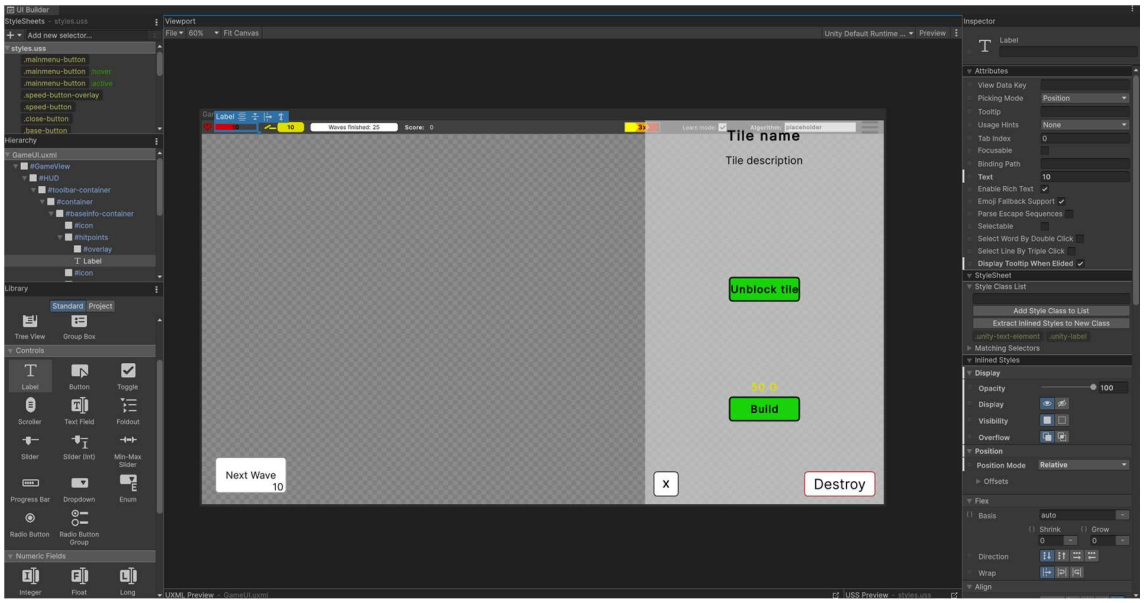
Pomocí skriptů lze snadno přistupovat ke GameObjectům a jejich komponentám ve scéně pomocí funkcí „GameObject.Find“ nebo „GameObject.FindGameObjectWithTag“. Pro přístup ke komponentám lze využít metodu „GetComponent“, která umožňuje získat odkaz na konkrétní komponentu připojenou ke GameObjectu. Poté lze manipulovat vlastnostmi a funkcemi těchto komponent, pro ovlivnění chování objektů ve scéně.

5.2.4.4 Události a interakce mezi objekty

Unity umožňuje vytvářet interakce mezi objekty prostřednictvím událostí a posluchačů. Události mohou být vyvolány skriptem v reakci na různé podmínky nebo akce, jako jsou kolize, uživatelské vstupy nebo časovače. Posluchači („listeners“) jsou skripty nebo metody, které jsou registrovány k odběru těchto událostí a provádějí specifické akce, když jsou události vyvolány.

5.2.5 UI v Unity

V Unity je jedním z klíčových nástrojů pro tvorbu uživatelského rozhraní (UI) UI Builder, neboli UI Toolkit. Tento nástroj usnadňuje vývojářům tvorbu interaktivních a responzivních UI prvků pro jejich hry a aplikace a díky snadnému drag-and-drop systému mohou vývojáři rychle navrhovat a upravovat UI prvky přímo ve scéně. UI Builder poskytuje širokou škálu předdefinovaných UI prvků, jako jsou tlačítka, posuvníky, textová pole, panely, obrazovky, přepínače a mnoho dalších prvků. Tyto prvky mohou být snadno přizpůsobeny a začleněny do scén dle potřeb vývojáře. Prvky mohou být také uspořádány do hierarchie, což umožňuje vytvářet komplexní a strukturovaná uživatelská rozhraní, která zahrnují vnořené panely, záložky a další prvky. Klíčovou vlastností UI Builderu je podpora stylování, která umožňuje vytvářet globální styly, které poté lze aplikovat na různé UI prvky, což zajišťuje konzistentní vzhled a chování napříč celým uživatelským rozhraním. Mimo jiné UI Builder také zahrnuje podporu pro animace UI prvků, které lze spouštět v reakci na události jako je například hover.



Obrázek 7 UI Builder v Unity (zdroj: autor)

5.2.6 Optimalizace a export projektu

Optimalizace a ladění výkonu je klíčové pro vytvoření plynulého a příjemného zážitku pro hráče. Unity poskytuje nástroje pro sledování výkonu, jako je Profiler, který umožňuje vývojářům analyzovat výkon svých her a identifikovat problémy. Vývojáři by měli optimalizovat své projekty pro různé platformy a zařízení, což zahrnuje snižování zátěže na CPU a GPU, efektivní správu paměti, optimalizaci fyziky a AI, a minimalizaci počtu vykreslovaných objektů. Tyto optimalizace zlepšují celkový výkon hry a zajišťují hladký chod na široké škále zařízení. Při exportu projektu v Unity je důležité správně nastavit platformu a build. Unity nabízí možnost buildu pro širokou škálu platforem, jako jsou PC, konzole, mobilní zařízení nebo webové prohlížeče.

5.3 *Co je to tower defense*

Tower defense je žánr strategických videoher, který se zaměřuje na obranu hráčova území před vlnami nepřátelských jednotek. Hráči musí umisťovat obranné věže (angl. „tower“) na strategických místech na herní ploše, aby ubránil (angl. „defense“) svou základnu před nepřátelskými jednotkami. Herní plocha je obvykle rozdělena na cesty, po kterých se pohybují nepřátelské jednotky a na volná místa určená pro stavbu věží. Hráč musí, s ohledem na trasu, po které se nepřátelské

jednotky pohybují, strategicky rozmisťovat věže, aby maximalizoval jejich účinek a škody, které způsobí. Většinou hra disponuje více typy věží, které se liší nejen cenou, rychlostí střelby a dosahem, ale třeba i schopnostmi a typem útoku, což vede k tomu, že hráči musí věže kombinovat, aby účinně čelili různým typům nepřátel. Hráči během hry mohou své věže vylepšovat, čímž zvyšují jejich účinnost a schopnost čelit silnějším nepřátelským jednotkám. Vylepšení mohou zahrnovat zvýšení rychlosti střelby, dosahu nebo škody. Nepřátelské jednotky se obvykle pohybují ve vlnách, které postupně zvyšují svou obtížnost, a obvykle existuje několik typů jednotek, které se liší svou rychlostí, množstvím životů a odolností vůči určitým typům útoků. Hráči ničením nepřátelských jednotek získávají herní měnu (např. zlato) a tuto měnu mohou použít ke koupi nových věži nebo k vylepšení těch stávajících. Správné hospodaření s měnou je klíčové pro úspěch ve hře.

Tower defense hry nabízejí širokou škálu strategických možností a výzev, které hráče nutí přemýšlet a plánovat své akce. Díky svému zaměření na strategii a rozhodování jsou tyto hry vhodné pro výuku a procvičování kritického myšlení a řešení problémů.

6 Aplikace

V následujících kapitolách je popsané vytváření samotné hry „Kingdom Defender“. Hra je dostupná ke stažení na adrese https://thatsnotmecz.itch.io/kingdom-defender. Návod na spuštění a uživatelskou příručku je možné nalézt v Příloze A.

6.1 *Požadavky*

Aplikace by pro podporu výuky diskrétní matematiky měla zahrnovat následující prvky:

- Herní plocha by měla být navržena, tak aby reprezentovala graf.
- Do hry by měly být začleněny algoritmy z teorie grafů, jako je hledání nejkratší cesty.
- Hra by měla hráče nabádat k interaktivnímu využití znalostí z teorie grafů, například možností modifikovat strukturu grafu a upravovat vlastnosti věží.
- Hra by měla disponovat intuitivním uživatelským rozhraním a ovládáním a měla by disponovat zvukovými a vizuálními efekty, aby udržela zájem hráčů.

6.2 *Grafické uživatelské rozhraní*

Grafické uživatelské rozhraní (*GUI*) hraje klíčovou roli v tom, jak uživatelé vnímají a používají aplikaci, proto je důležité, aby bylo intuitivní, přístupné a esteticky zaujalo. Vzhledem k tomu, že praktická část je ve fázi prototypu, tak můžeme předpokládat, že projekt nebude splňovat poslední bod, kterým je estetická přitažlivost GUI.

6.3 *Reprezentace grafu*

Graf je ve hře reprezentován pomocí mapy a jejích dlaždic. Každou dlaždici „*Tile*“ na mapě lze z vizuálního hlediska považovat za vrchol grafu, který je identifikován pomocí souřadnic ***x*** a ***y***. Každá dlaždice také obsahuje informaci o krajině, zda se jedná o počáteční nebo cílový bod trasy nepřátelských jednotek.

Každé dlaždici mapy odpovídá uzel „*Node*“ z mřížky uzlů „grid“, jedná se o datovou strukturu „Dictionary“ neboli slovník, kde se jako klíč používá souřadnice dlaždice typu „Vector2Int“ a hodnotou je již zmíněný „Node“. Právě zmíněný „grid“ je využíván pro hledání cest v grafu, hrany mezi uzly jsou implicitně definovány jako spojení mezi sousedními uzly, které mají společnou stranu. Náklady na pohyb, se kterými počítá například Dijkstrův algoritmus jsou reprezentovány pomocí vlastnosti „cost“ ve třídě „Node“. Vlastnost „cost“ je vypočítávána na základě modifikátorů krajiny, která se nachází na dané dlaždici.

6.3.1 Generování mapy

Veškeré generování mapy probíhá pomocí pseudonáhodného generování na základě seedu uvedeného ve třídě „Seed“. Nejprve se vytvoří prázdná mřížka dlaždic o velikosti „xTiles“ a „yTiles“, kde každá dlaždice je instancí třídy „Tile“. Po vytvoření dlaždic je na každé z nich nastaven typ krajiny, jako je řeka, bažina, les nebo louka a umístí se počáteční a koncový bod, případně vybraný počet checkpointů (kontrolních bodů) na mapě, které jsou využity v algoritmu obchodního cestujícího. Nakonec se inicializuje „vybarvení“ mapy na základě krajiny na každé dlaždici.

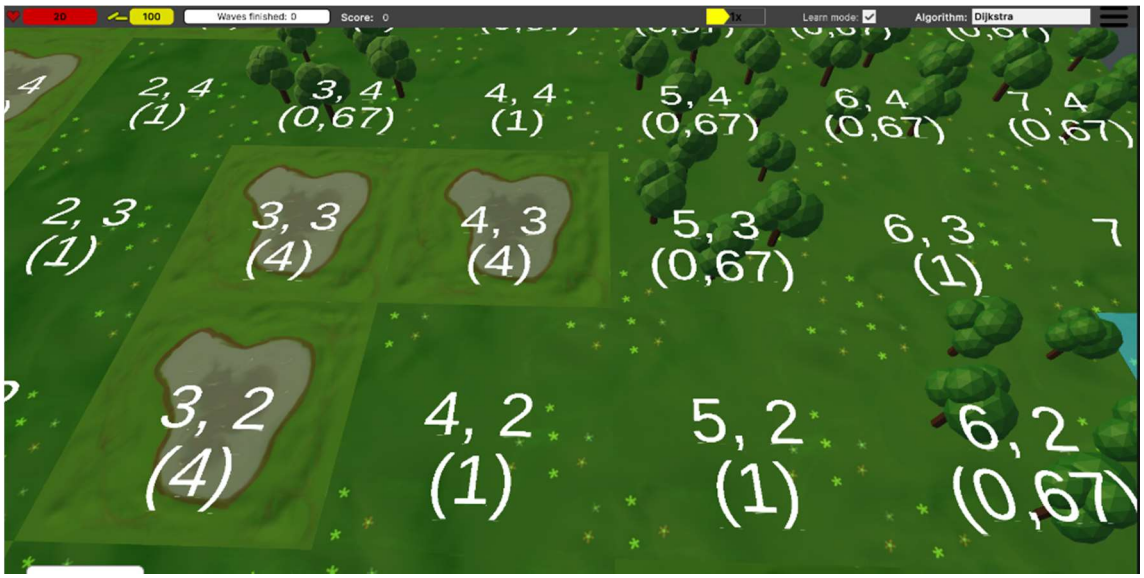
Tento algoritmus poskytuje flexibilní základ pro vytváření různorodých scénářů pro hru a lze ho snadno rozšířit o další typy krajin, které by mohly být využitelné pro další algoritmy ve výuce diskrétní matematiky.

6.3.2 Ohodnocení vrcholů

Každý vrchol grafu představuje jedno políčko mapy. Vrcholy jsou ohodnoceny podle několika parametrů, které ovlivňují chování jednotek a interakce s prostředím. Náročnost přechodu přes daný vrchol je reprezentována parametrem „cost“, který se pro každé políčko vypočítává na základě krajiny, která se na něm nachází. Krajina („Landscape“) má nastaveny hodnoty „SpeedMultiplier“ a „ArmorMultiplier“, které ovlivňují rychlost a obranyschopnost nepřátelských jednotek. Na základě těchto parametrů se vypočítává celková cena přechodu přes vrchol „totalCost“.

```
float speedCost = 1 / SpeedMultiplier;
float defenseCost = 1 / ArmorMultiplier;
float totalCost = speedCost * defenseCost;
```

Ukázka kódu 5 Výpočet ceny pohybu přes vrchol grafu (zdroj: autor)



Obrázek 8 Vizuálně znázorněné ohodnocení políček mapy (zdroj: autor)

6.3.3 Algoritmy pro hledání cest

Tato kapitola se zaměřuje na implementaci dříve zmíněných algoritmů v praktické části práce.

6.3.3.1 Prohledávání do šířky

Prohledávání do šířky je algoritmus pro prohledávání grafu, který od počátečního vrcholu po vrstvách zkoumá všechny vrcholy ve stejné vzdálenosti od počátečního vrcholu dříve, než přejde na další vrstvu. Pro prohledávání je použita fronta „frontier“, která udržuje vrcholy k prozkoumání. Algoritmus začíná přidáním startovního vrcholu do fronty a pokračuje prohledáváním sousedních vrcholů pomocí metody „ExploreNeighborsBFS“. Sousední vrcholy, které dosud nebyly prozkoumány a jsou schůdné, jsou přidány do fronty. Proces pokračuje, dokud není nalezen cílový vrchol nebo fronta není prázdná.

```
void BreadthFirstSearch(Vector2Int startCoords) {
    ...
    // Počáteční vrchol přidáme do fronty pro zpracování (frontier)
    frontier.Enqueue(grid[startCoords]);
    reached.Add(startCoords, grid[startCoords]);
    ...
    //  Každý  vrchol  z frontier  postupně  odstraníme  a  prozkoumáme  jeho
    //  sousedy, dokud nenajdeme vrchol, který je cílový
    ExploreNeighborsBFS();
    ...
}

void ExploreNeighborsBFS() {
  List<Node> neighbors = new List<Node>();

  // Procházíme všechny směry a zjišťujeme souřadnice sousedních vrcholů
  foreach (Vector2Int direction in directions) {
    Vector2Int neighborCoords = currentSearchNode.coordinates + direction;

    // Pokud sousední vrchol existuje, přidáme jej do seznamu
    if (grid.ContainsKey(neighborCoords)) {
      neighbors.Add(grid[neighborCoords]);
    }
  }

  // Procházíme všechny sousední vrcholy
  foreach (Node neighbor in neighbors) {
    if(
     !reached.ContainsKey(neighbor.coordinates)
      && neighbor.isWalkable
    ) {
          // Pokud sousední vrchol nebyl dosud prozkoumán a je schůdný
             nastavíme aktuální vrchol jako vrchol, ze kterého jsme přišli
          neighbor.connectedTo = currentSearchNode;
          reached.Add(neighbor.coordinates, neighbor);

          // sousední vrchol přidáme do fronty pro další prozkoumávání
          frontier.Enqueue(neighbor);
     }
   }
}
```

Ukázka kódu 6 Implementace prohledávání do šířky (zdroj: autor)

6.3.3.2 Dijkstrův algoritmus

Dijkstrův algoritmus je algoritmus pro nalezení nejkratší cesty v grafu s nezápornými váhami hran. V kódu je implementován v metodě „*Dijkstra(start, destination)*“. Algoritmus používá frontu „*frontier*“ k udržování vrcholů k prozkoumání a slovník „*reached*“ pro ukládání dosažených vrcholů. Během průchodu algoritmus zkoumá sousedy aktuálního vrcholu a aktualizuje náklady na jejich dosažení. Metoda vrací cenu nejkratší cesty mezi startovním a cílovým vrcholem, která je využívána primárně při výpočtu algoritmu obchodního cestujícího. Implementaci algoritmu lze popsat v následujících bodech:

1. Inicializace: Na začátku jsou inicializovány datové struktury, jako jsou fronta (frontier) a slovník (reached). Nastavíme počáteční uzel (start) s nákladem 0 a vložíme ho do fronty.
2. Prozkoumávání grafu: Algoritmus pokračuje tím, že opakovaně prochází graf, dokud fronta není prázdná. V každém kroku se vybere uzel s nejnižším nákladem z fronty a zpracuje se. Pokud je tento uzel cílový, algoritmus končí.
3. Zpracování sousedních uzlů: Pro každý zpracovávaný uzel zjistíme seznam sousedních uzlů a seřadíme je podle jejich ceny. Poté pro každého souseda kontrolujeme, zda je průchodný. Pokud ano, spočítáme celkový náklad na dosažení tohoto souseda přes aktuální uzel.
4. Aktualizace nákladů a předchozích uzlů: Pokud je nově spočítaná cena nižší než dosavadní cena k dosažení souseda, aktualizujeme jeho cenu. Pokud je soused cílový uzel a jeho náklad je nižší než dosavadní nejnižší celková cena, aktualizujeme celkovou cenu.
5. Přidání sousedů do fronty: Pokud soused není cílový uzel, tak je přidán do fronty pro další zpracování, které probíhá podle bodů 2 až 4.

6.3.3.3 Problém obchodního cestujícího

Problém obchodního cestujícího spočívá v nalezení nejkratší možné cesty mezi několika městy, která navštíví každé město právě jednou a vrátí se na počátek. V této hře jsou města reprezentována kontrolními body (angl. checkpoints)

a počáteční a koncový bod není totéž pole, ale jedná se o hráčovu a nepřátelskou základu. Implementace algoritmu pro řešení problému obchodního cestujícího vypadá následovně:

1. Inicializace: Nejprve nastavíme počáteční a cílové souřadnice, celkovou cenu cesty na nekonečno a vytvoříme seznam všech možných permutací pořadí navštívení kontrolních bodů.
2. Procházení permutací: Pro každou permutaci kontrolujeme celkovou cenu cesty. Nejprve vytvoříme seznam kontrolních bodů včetně počátečních a cílových souřadnic a následně procházíme každou dvojici sousedních měst.
3. Výpočet nákladů: Pro každou dvojici sousedních měst vypočítáme náklady na cestu mezi nimi pomocí implementace Dijkstrova algoritmu popsané v kapitole 6.3.3.2. Pokud jsou celkové náklady vyšší než dosavadní nejnižší náklady, přeskočíme tento krok a pokračujeme v dalších permutacích.
4. Aktualizace nejlepší cesty: Pokud je celkové náklady nižší než dosavadní nejnižší náklady, aktualizujeme nejnižší náklady a uložíme aktuální pořadí bodů jako nejoptimálnější cestu.
5. Sestavení celkové cesty: Po prozkoumání všech permutací sestavíme celkovou cestu podle nejoptimálnějšího pořadí měst. Pro každou dvojici sousedních měst v nejlepším pořadí získáme nejkratší cestu pomocí Dijkstrova algoritmu a přidáme ji do celkové cesty.

6.4 *Herní objekty a mechaniky*

6.4.1 **Věže a jejich vlastnosti**

Tato kapitola se zaměřuje na věže a jejich vlastnosti vytvořené pomocí Blenderu a Unity, i když v současném prototypu je k dispozici pouze věž "Balista". Věže mají několik společných proměnných a vlastností, které určují jejich chování a schopnosti, a tyto parametry jsou stejné pro všechny věže.

Mezi útočné schopnosti patří základní údaj o poškození, rychlost střelby, rychlost střel a dosah. Tyto hodnoty ovlivňují, jak efektivní je věž při útocích na nepřátele. Věže mají také cenu v herní měně, která určuje, kolik hráč musí zaplatit za její umístění na herním plánu.

Zkušenosti a úroveň věže jsou dalšími důležitými aspekty. Věže získávají zkušenosti za ničení nepřátel a postupují na vyšší úrovně, které zlepšují její vlastnosti. Každá úroveň vyžaduje určité množství zkušeností k dosažení, a věže mohou dosáhnout maximálního úrovně.

Hráč může vylepšovat schopnosti věže, což zlepší její výkonnost v boji. Vylepšení zahrnují zvýšení poškození, dosahu a rychlosti střelby. Každé vylepšení má svou cenu a násobitel, který určuje, jak moc se daná vlastnost zlepší. Ačkoli v prototypu je pouze věž "Balista", tyto parametry a vlastnosti jsou univerzální a mohou být aplikovány na všechny věže ve hře. Díky těmto společným proměnným a vlastnostem může hráč upravovat schopnosti věží a sledovat jejich výkonnost, což z nich činí klíčové prvky ve hře.

6.4.2 **Nepřátelé a jejich atributy**

Nepřátelské jednotky mají řadu proměnných a vlastností, které ovlivňují jejich chování a schopnosti během hry. Každá nepřátelská jednotka má maximální počet životů (*maxHP*) a aktuální počet životů (*currentHP*). Tyto hodnoty určují, jak odolná je jednotka proti útokům hráčových věží. Navíc má každá jednotka základní obrannou hodnotu (*baseArmor*) a aktuální obrannou hodnotu (*CurrentArmor*), které zohledňují bonusy z krajiny, na které jednotka stojí.

Jednotky postupují podél cesty a jejich cílem je dosáhnout hráčovy základny. Rychlost pohybu jednotek (*baseMovementSpeed* a *currentMovementSpeed*) je také ovlivněna krajinou, na které se jednotka nachází. Když jednotka utrpí poškození od hráčovy věže, je volána funkce, která snižuje její aktuální počet životů. Pokud je jednotka zničena, hráč obdrží herní měnu (*moneyForKill*) za její zabití. Kromě toho mají nepřátelské jednotky vlastnosti jako „spawnRate“, která určuje frekvenci, s jakou se jednotky objevují ve hře, a „minDiff“, který určuje minimální obtížnost, na které se jednotka může objevit. Nepřátelské jednotky jsou klíčovým prvkem ve hře, který hráč musí překonat prostřednictvím strategického umístění a upgradování věží.

6.5 *Hlavní herní smyčka*

6.5.1 **Spouštění vln nepřátel**

V rámci prototypu hry, který neobsahuje konečné vítězství, se vytváří nekonečný počet náhodných vln nepřátel. Tento proces je řízen třídou WaveGenerator a WaveManager.

WaveGenerator je zodpovědný za generování náhodných vln nepřátel. Obtížnost hry se zvyšuje v průběhu času, což je řízeno proměnnou _difficulty. Vlny jsou generovány pomocí metody GenerateRandomWave(), která využívá hodnotu obtížnosti k určení počtu nepřátel a jejich modifikátoru životů.

V třídě WaveManager je smyčka vytváření vln implementována prostřednictvím coroutines. Tato smyčka je nekonečná a pokračuje, dokud hráč neprohraje. Během smyčky se kontroluje, zda je možné spustit další vlnu nepřátel. Pokud ano, je vygenerována náhodná vlna nepřátel pomocí WaveGeneratoru a spuštěna metoda SpawnEnemy(), která postupně vytváří nepřátele ve vlně s danými parametry.

Cílem hráče v tomto prototypu hry je dosáhnout co nejvyššího skóre, které je dosaženo porážkou co nejvíce nepřátel. Obtížnost hry se postupně zvyšuje s počtem vygenerovaných vln, aby se přizpůsobila fázi hry a hra byla neustálou výzvou.

6.5.2 **Balancování obtížnosti hry**

Obtížnost hry se postupně zvyšuje v průběhu času prostřednictvím třídy „WaveGenerator“. Obtížnost je řízena proměnnou „_difficulty“, která je zvýšena při každém dokončení vlny o hodnotu „difficultyIncreasePerWaveComplete“. Dále je obtížnost použita při generování náhodných vln nepřátel v metodě „GenerateRandomWave()“, kde ovlivňuje počet nepřátel ve vlně a jejich zdravotní modifikátor. Tímto způsobem se hra stává postupně těžší a výzvy se přizpůsobují pokroku hráče.

7 Testování aplikace

7.1 *Úvod do uživatelského testování*

Cíle uživatelského testování spočívají ve zjištění, zda je hra efektivní jako podpora výuky teorie grafů a zda vzbudí větší zájem studentů o daný předmět. Testování také umožní identifikovat silné stránky a slabá místa hry, aby mohla být aplikace dále zlepšována. Uživatelské testování má velký význam pro vývoj a zdokonalení aplikace, neboť pomáhá pochopit potřeby a očekávání uživatelů a identifikovat oblasti, které je třeba zlepšit nebo upravit. Kritéria úspěchu zahrnují zlepšení zájmu studentů o teorii grafů, zvýšení motivace k učení látky související s teorií grafů. Testovací scénář bude zahrnovat vyzkoušení hry studenty, kteří studují či studovali předměty spojené s teorií grafů. Po vyzkoušení hry vyplní studenti krátký dotazník, který poskytne zpětnou vazbu o jejich zkušenostech s hrou a jejím dopadu na jejich zájem o dané předměty.

Očekávané výsledky zahrnují zvýšení zájmu studentů o teorii grafů a související předměty, vylepšení hry na základě zpětné vazby od studentů a případné návrhy na další rozšíření nebo vylepšení hry, aby byla lépe využitelná jako podpůrná aplikace pro výuku teorie grafů.

Dotazník je k dispozici v Příloze B.

7.2 *Vyhodnocení otázek dotazníku*

Na základě odpovědí z dotazníku lze vyhodnotit následující aspekty hry:

1. Ovládání hry: Většina respondentů považuje ovládání hry za intuitivní. Nicméně někteří uživatelé navrhli úpravy, jako například změnu pohybu kamery, lepší zobrazení dosahu věží při stavbě nebo rychlejší přibližování kolečkem myši.
2. Výukový potenciál hry: Většina respondentů vnímá hru jako dobrou podporu pro výuku předmětů vyučujících teorii grafů. Někteří uživatelé navrhli více algoritmů nebo vylepšení zobrazení aktuální cesty.
3. Motivace a zájem o předměty: Respondenti měli smíšené pocity ohledně toho, jak hra ovlivnila jejich motivaci k učení a zájem o dané předměty.

Někteří uživatelé uvedli, že hra by zvýšila jejich zájem, zatímco jiní neměli žádný názor nebo jejich motivace nebyla ovlivněna.

Aspekt	Počet pozitivních odpovědí	Počet neutrálních odpovědí	Počet negativních odpovědí	Návrhy na zlepšení
Ovládání hry	7	0	4	Změna ovládání kamery, lepší zobrazení dosahu věží, citlivější zoom
Výukový potenciál	9	2	0	Nabídnout více algoritmů, vylepšit zobrazení aktuální cesty
Motivace a zájem	6	3	2	-

Tabulka 1 Uživatelské testování

Součástí dotazníku byl také dotaz na frekvenci, s jakou respondent hraje počítačové hry (denně, 3-5x týdně, jednou týdně nebo méně často). Nicméně ze získaných informací není možné určit přímý vztah mezi frekvencí hraní PC her a odpověďmi na dotazník, ale lze říci, že respondenti, kteří hrají PC hry méně často, mají více potíží s ovládáním hry. Výukový potenciál hry a motivace k učení a zájem o předměty jsou smíšené napříč všemi skupinami respondentů.

7.3 *Závěr uživatelského testování*

Uživatelské testování ukázalo, že většina uživatelů považuje ovládání hry za intuitivní, i když někteří měli návrhy na zlepšení, jako je pohyb kamery, citlivost otáčení, zobrazení dosahu věží nebo rychlost přibližování. Někteří uživatelé uvedli, že hra zvýšila jejich zájem a motivaci ke studiu, zatímco jiní měli na tento aspekt neutrální nebo žádný názor. Uživatelé navrhli řadu možných vylepšení pro hru, jako je výraznější zobrazení aktuální cesty, implementování více algoritmů z teorie grafů nebo přidání nápovědy popisující grafické rozhraní, ovládání a jednotlivé algoritmy. Nebyl nalezen žádný významný vztah mezi frekvencí hraní her a odpověďmi respondentů na další otázky dotazníku. Celkově lze říci, že prototyp hry má

potenciál být účinnou podporou výuky předmětů zabývajících se teorií grafů, ale je třeba zvážit některé z návrhů uživatelů pro další vývoj a vylepšení.

8 Závěry a doporučení

V této závěrečné kapitole práce jsou shrnuty klíčové poznatky a dosažené výsledky vývoje a testování prototypu počítačové hry "Kingdom Defender". Hra byla vytvořena v softwarových nástrojích Unity a Blender s cílem podpořit výuku teorie grafů. Důraz byl kladen na implementaci algoritmů pro hledání cest, jako jsou prohledávání do šířky, Dijkstrův algoritmus a řešení problému obchodního cestujícího, které byly detailně popsány v práci společně s vlastní implementací.

V rámci práce byly také analyzovány další hry, které využívají prvky teorie grafů, jako jsou Stellaris, Civilization a Cities: Skylines. Nicméně tyto hry nejsou určeny pro podporu výuky a slouží primárně jako reference na obílené hry s těmito prvky. Tento přehled ukázal, že výuka prostřednictvím her má potenciál být efektivní a zábavná metoda pro usnadnění pochopení složitých konceptů a prohloubení znalostí studentů.

Výsledky testování odhalily, že hra je intuitivní a zvýšila by zájem některých studentů o teorii grafů. Během testování poskytli respondenti cennou zpětnou vazbu a navrhli několik možných vylepšení pro další vývoj a zdokonalení hry. Tato odezva je klíčová pro úspěšné zlepšení a dosažení plného výukového potenciálu.

Prototyp "Kingdom Defender" má na základě získaných informací potenciál stát se užitečným nástrojem pro podporu výuky teorie grafů a souvisejících předmětů. Pro dosažení tohoto cíle je nezbytné pečlivě zohlednit názory studentů a provést nezbytná vylepšení a úpravy hry. V budoucích krocích je důležité zaměřit se na rozšíření a zdokonalení hry, aby se stala ještě atraktivnějším a efektivnějším nástrojem pro výuku.

Tato práce představuje inovativní přístup k výuce teorie grafů prostřednictvím herního prostředí a ukazuje potenciál, který takové projekty mají pro zlepšení výuky a podpory studentů při učení. Jejím cílem je inspirovat další výzkumy a projekty v oblasti výuky diskrétní matematiky a dalších předmětů, a tím přispět ke zkvalitnění vzdělávání.

Pro další vývoj a úspěšné začlenění hry do výukových procesů by byla nezbytná spolupráce s pedagogy a odborníky z oboru teorie grafů. Je důležité vytvořit vhodný výukový materiál, který bude doplňovat herní zkušenost a poskytovat studentům

nejen praktické, ale i teoretické znalosti. Pro další zkoumání by bylo zajímavé provést rozšířenější uživatelské testování s větším počtem respondentů, aby bylo možné získat širší spektrum názorů a zjistit, jak se hra osvědčí v různých vzdělávacích kontextech. To by mohlo pomoci identifikovat nové možnosti pro další rozvoj a zlepšení hry, aby byla co nejefektivnější a co nejzábavnější.

V neposlední řadě je třeba zmínit, že vývoj takovýchto výukových nástrojů, jako je "Kingdom Defender", je neustálý proces. Průběžně se objevují nové technologie a pedagogické metody, které by mohly být využity pro zlepšení výuky. Je proto důležité neustále sledovat nové trendy a inovace a zvažovat jejich implementaci do výukových nástrojů, aby se tyto nástroje stále zdokonalovaly a držely krok s rychle se měnícím světem vzdělávání.

Tato práce by mohla představovat důležitý krok směrem k vytvoření plně funkčního a komplexního výukového nástroje, který by mohl být široce používán ve výuce teorie grafů. Předkládaná práce tedy přispívá ke zkvalitnění vzdělávání a zvyšuje motivaci studentů v oblasti teorie grafů a s tím souvisejících předmětů.

9 Seznam použité literatury

[1] WOLF, Mark J. P. *The Medium of the Video Game*. B.m.: University of Texas Press, 2002. ISBN 978-0-292-79150-3.

[2] DONOVAN, Tristan a Richard GARRIOTT. *Replay: The History of Video Games*. B.m.: Yellow Ant, 2010. ISBN 978-0-9565072-0-4.

[3] KENT, Steven L. *The Ultimate History of Video Games, Volume 1: From Pong to Pokemon and Beyond . . . the Story Behind the Craze That Touched Our Lives and Changed the World*. B.m.: Crown, 2010. ISBN 978-0-307-56087-2.

[4] PLASS, Jan L, Richard E MAYER a Bruce D HOMER. Handbook of Game-Based Learning. *MIT Press* [online]. [vid. 2023-04-18]. Dostupné z: https://mitpress.mit.edu/9780262043380/handbook-of-game-based-learning/

[5] MAYER, Richard E. Computer Games in Education. *Annual Review of Psychology* [online]. 2019, **70**(1), 531–549. Dostupné z: doi:10.1146/annurev-psych-010418-102744

[6] This War of Mine. *11 bit studios* [online]. [vid. 2023-04-18]. Dostupné z: https://11bitstudios.com/games/this-war-of-mine/

[7] PUBLISHED, Dustin Bailey. The Polish government is providing free copies of This War of Mine to high school students. *gamesradar* [online]. 28. červen 2022 [vid. 2023-04-18]. Dostupné z: https://www.gamesradar.com/the-polish-government-is-providing-free-copies-of-this-war-of-mine-to-high-school-students/

[8] *Polish High Schools Adding This War of Mine to Curriculum* [online]. [vid. 2023-04-18]. Dostupné z: https://gamerant.com/this-war-of-mine-game-polish-high-schools/

[9] RAHMAN, Md. Saidur. *Basic Graph Theory* [online]. Cham: Springer International Publishing, 2017 [vid. 2023-01-16]. Undergraduate Topics in Computer Science. ISBN 978-3-319-49474-6. Dostupné z: doi:10.1007/978-3-319-49475-3

[10] CORMEN, Thomas H., ed. *Introduction to algorithms*. 3rd ed. Cambridge, Mass: MIT Press, 2009. ISBN 978-0-262-03384-8.

[11] MEHLHORN, Kurt a Peter SANDERS. *Algorithms and Data Structures* [online]. Berlin, Heidelberg: Springer, 2008 [vid. 2023-03-12]. ISBN 978-3-540-77977-3. Dostupné z: doi:10.1007/978-3-540-77978-0

[12] BARBEHENN, Michael. A note on the complexity of Dijkstra's algorithm for graphs with weighted vertices. *Computers, IEEE Transactions on* [online]. 1998, **47**, 263. Dostupné z: doi:10.1109/12.663776

[13] CHOI, In Chan, Seong-in KIM a Hak-Soo KIM. A genetic algorithm with a mixed region search for the asymmetric traveling salesman problem. *Computers &*

Operations Research [online]. 2003, **30**, 773–786. Dostupné z: doi:10.1016/S0305-0548(02)00050-3

[14] ZHI, X.H., X.L. XING, Q.X. WANG, L.H. ZHANG, X.W. YANG, C.G. ZHOU a Y.C. LIANG. A discrete PSO method for generalized TSP problem. *Proceedings of 2004 International Conference on Machine Learning and Cybernetics (IEEE Cat. No.04EX826)* [online]. 2004, **4**, 2378–2383. Dostupné z: doi:10.1109/ICMLC.2004.1382200

[15] BENGIO, Yoshua, Andrea LODI a Antoine PROUVOST. Machine learning for combinatorial optimization: A methodological tour d'horizon. *European Journal of Operational Research* [online]. 2021, **290**(2), 405–421. ISSN 0377-2217. Dostupné z: doi:10.1016/j.ejor.2020.07.063

[16] SCHADD, M.P.D. *Selective search in games of different complexity* [online]. B.m., 2011 [vid. 2023-04-23]. maastricht university. Dostupné z: doi:10.26481/dis.20110525ms

[17] Civilization® VI – The Official Site. *Civilization® VI – The Official Site* [online]. [vid. 2023-03-28]. Dostupné z: https://civilization.com/

[18] *Cities: Skylines - Paradox Interactive* [online]. [vid. 2023-03-28]. Dostupné z: https://paradoxinteractive.com/games/cities-skylines/about

[19] *Stellaris - Paradox Interactive* [online]. [vid. 2023-03-28]. Dostupné z: https://paradoxinteractive.com/games/stellaris/about

[20] FOUNDATION, Blender. blender.org - Home of the Blender project - Free and Open 3D Creation Software. *blender.org* [online]. [vid. 2023-02-26]. Dostupné z: https://www.blender.org/

[21] *Autodesk FBX SDK Documentation: What is Autodesk FBX Technology?* [online]. [vid. 2023-03-31]. Dostupné z: http://docs.autodesk.com/FBX/2014/ENU/FBX-SDK-Documentation/index.html?url=files/GUID-274163DA-9E89-4DCC-8AF6-10B0C498582E.htm,topicNumber=d30e206

[22] *Autodesk empowers innovators everywhere to make the new possible* [online]. [vid. 2023-03-31]. Dostupné z: https://www.autodesk.com/

[23] TECHNOLOGIES, Unity. *Unity Real-Time Development Platform | 3D, 2D VR & AR Engine* [online]. [vid. 2023-02-27]. Dostupné z: https://unity.com/

Přílohy

Příloha A

Uživatelská příručka

1. Úvod

Hra „Kingdom Defender“ slouží pro podporu výuky teorie grafů a zvýšení zájmu studentů o předměty, které se touto látkou zabývají. Jedná se o prototyp tower defense hry s nízkými hardwarovými požadavky a snadným ovládáním.

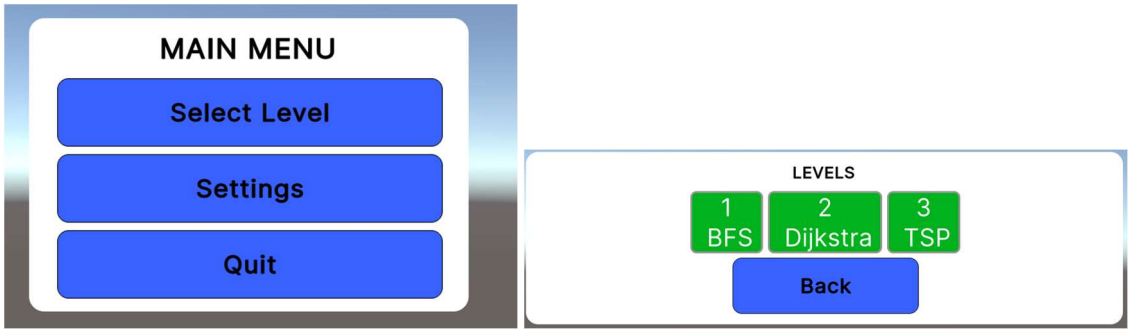
2. Požadavky na systém

Aktuální verze podporuje pouze systémy Windows, a jak již bylo zmíněno, tak hardwarové nároky jsou minimální a lze ji spustit prakticky na jakémkoliv počítači s klávesnicí a myší s kolečkem.

3. Instalace a spuštění

Hra je dostupná ke stažení na adrese https://thatsnotmecz.itch.io/kingdom-defender. Hru pro spuštění není třeba instalovat. Spouští se pomocí „Kingdom Defender.exe“ v kořenové složce. Pro odstranění zbytkových souborů je třeba brát na vědomí, že nastavení hry (aktuálně pouze nastavení zvuku) je ukládáno do složky „%userprofile%/documents/My Games/Tower Defense“.

4. Rozhraní aplikace

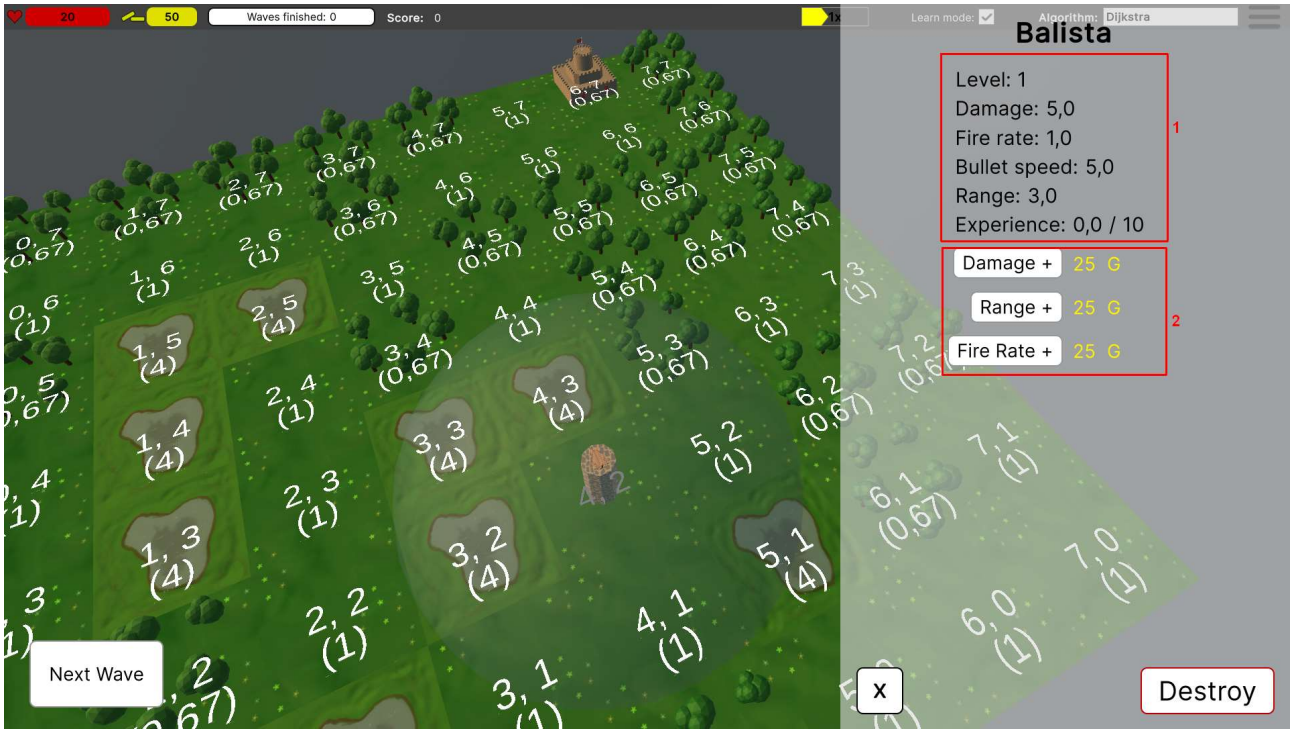


Na obrázcích výše lze vidět hlavní menu (vlevo) s možnostmi přejít na výběr kola (vpravo), změnit nastavení a ukončit hru.

Na následujících obrázcích jsou označeny důležité části uživatelského rozhraní a hry samotné a následně jsou tyto body popsány.

1. Počet životů hráčova hradu, pokud klesne na 0, hra končí.
2. Hráčovy zlaťáky, potřebné pro stavbu a vylepšování věží. Získávané za ničení nepřátelských jednotek.
3. Počet vln nepřátel, které opustily hrad nepřítele.
4. Aktuální skóre, body jsou získávány za každého zničeného nepřítele.
5. Nastavení rychlosti hry – 1x, 2x nebo 3x.
6. Zapínání výukového režimu.
7. Algoritmus, který je v daném kole použitý pro výpočet cesty nepřátel.
8. Tlačítko pro zobrazení menu a pozastavení hry.
9. Tlačítko pro vynucení další vlny nepřátel. Ve „výukovém“ režimu je třeba vlny spouštět manuálně, v opačném případě se další vlna nepřátel spustí po uplynutí přednastaveného časového limitu.
10. Hráčův hrad. Cílem je ho ubránit tak dlouho, jak je to možné.
11. Nepřátelský hrad, odtud chodí nepřátelské jednotky.

1. Zvýraznění vybraného pole na mapě.
2. Výběr věže k postavení.
3. Počet zlaťáků potřebných pro postavení věže.
4. Tlačítko pro potvrzení stavby věže.
5. Tlačítko pro zničení věže, pokud je na daném poli postavená.



1. Informace o věži postavené na vybraném poli.
2. Tlačítka pro vylepšení jednotlivých parametrů věže a jejich cena.

5. Základní ovládání hry

Pro využití všech funkcí je při hraní potřeba myš s kolečkem, čili na touchpadu je ovládání aktuálně mírně omezené.

- Po mapě se lze pohybovat pomocí kláves W (vpřed), A (vlevo), S (zpět), D (vpravo).
- Přiblížení a oddálení kamery se provádí pomocí kolečka.
- Rotace se provádí podržením kolečka myši.
- Hru lze pozastavit mimo jiné také pomocí klávesy escape.

6. Herní mechaniky

Při zapnutém výukovém režimu se na mapě zobrazují souřadnice polí a jejich náročnost na průchod. V tomto režimu je cílem hráče využít své znalosti teorie grafů a vypočítat, či odhadnout jakou trasu nepřátelé zvolí a přizpůsobit tomu strategii stavění věží.

Při vypnutém výukovém režimu se na mapě pomocí sfér (modrá barva) zobrazuje vypočítaná trasa, hráč tedy nemusí provádět žádné výpočty a může si pouze užívat hru nebo zkusit vysledovat, jakým způsobem výpočet trasy počítačem funguje.

V obou režimech platí, že pokud dojde k postavení věže na aktuálně vypočítané trase, tak dojde k jejímu přepočítání, v případě že je vypnutý výukový režim, tak se při výběru pole na trase zobrazí opět pomocí sfér (oranžová barva) nová trasa, kterou nepřátelské jednotky půjdou v případě postavení věže na tomto poli.

Pro hraní jsou k dispozici tři (3) kola/mapy. První je „BFS“, kde se pro výpočet tras používá prohledávání do šířky, druhou je „Dijkstra“, kde jak název napovídá, se pro výpočet trasy používá Dijkstrův algoritmus a posledním kolem je „TSP“, ve kterém jsou navíc také kontrolní body (žlutá pole), přes které musejí nepřátelské jednotky projít, než dorazí k hráčovu hradu. Cílem hráče je zvolit vhodnou strategii stavění a vylepšování věží pro získání co nejvyššího skóre.

Příloha B

Uživatelský dotazník

1. Jak často hrajete videohry?
   - Každý den
   - 3-5x týdně
   - Jednou týdně
   - Méně než jednou týdně

2. Je ovládání hry intuitivní?
   - Ano
   - Ne
     a. Pokud ne, co vám na ovládání nevyhovuje?

3. Jaký je váš obecný dojem z této hry jako podpory výuky předmětů DIMA a DMO?

4. Jaký dopad měla tato hra na vaši motivaci k učení a zájem o dané předměty?

5. Máte nějaké návrhy na vylepšení této hry, aby byla účinnější jako podpora výuky předmětu?

Univerzita Hradec Králové
Fakulta informatiky a managementu

Zadání diplomové práce

Autor:	**Bc. Jakub Schmidt**
Studium:	I2100079
Studijní program:	N1802 Aplikovaná informatika
Studijní obor:	Aplikovaná informatika
Název diplomové práce:	**Aplikace teorie grafů: Počítačová hra**
Název diplomové práce AJ:	Applications of graph theory: Computer game

Cíl, metody, literatura, předpoklady:

1. Rešerše dané problematiky
2. Natudování potřebné částí z teorie grafů a grafových algoritmů
3. Teoretické zpracování vybraného problému z teorie grafů
4. Výběr technologie pro tvorbů ICT nástroje (herního prostředku)
5. Vytvoření ICT nástroje
6. Popis ICT nástroje a jeho ovládání
7. Testování herního prostředku
8. Navržení rozvoje do budoucnosti

CORMEN, Thomas H., ed. *Introduction to algorithms*. 3rd ed. Cambridge, Mass: MIT Press, 2009. ISBN 978-0-262-03384-8.

RAHMAN, Md. Saidur. *Basic Graph Theory* [online]. Cham: Springer International Publishing, 2017 [vid. 2023-01-16]. Undergraduate Topics in Computer Science. ISBN 978-3-319-49474-6. Dostupné z: doi:10.1007/978-3-319-49475-3

Zadávající pracoviště:	Katedra informatiky a kvantitativních metod, Fakulta informatiky a managementu
Vedoucí práce:	RNDr. Andrea Ševčíková, Ph.D.
Datum zadání závěrečné práce:	1.2.2022