**PACMAN DOKUMENTACE**

**ÚVOD**

Program Pacman je implementací klasické hry. Kromě hraní tradičních levelů nabízí také integrovaný editor map, ve kterém si uživatel může stávající levely libovolně upravovat nebo vytvářet zcela nové, včetně vlastních obrázků a pozadí.

**PRAVIDLA HRY**

1. Hráč ovládá postavu Pacmana, který se pohybuje po mapě spolu s monstry, super monstry, sušenkami a super sušenkami.
2. Pohyb Pacmana se řídí pomocí kláves: W (nahoru), A (doleva), S (dolů), D (doprava). Pohyb monster je náhodný.
3. Cílem hry je sníst všechny sušenky a super sušenky, aniž by byl Pacman chycen a vyřazen monstry.
4. Pokud se Pacman dostane do kontaktu s monstrem nebo super monstrem, okamžitě prohrává a hra končí.
5. Po snědení super sušenky získá Pacman dočasnou sílu a vstoupí do takzvaného hunter módu. V tomto módu může při kontaktu zničit běžná monstra. Aktivní hunter mód je indikován šedým filtrem na obrazovce, který zároveň signalizuje zbývající dobu trvání módu.
6. Super monstra nelze zničit, a to ani v hunter módu.
7. Hra je úspěšně dokončena, pokud Pacman sežere všechny sušenky a super sušenky.

**UŽIVATELSKÁ DOKUMENTACE**

V této sekci se pokusím zaměřit na co nejlepší popis jednotlivých funkcionalit programu tak, aby vše bylo jasné nezasvěcenému uživateli. Po spuštění programu se zobrazí okno, které lze vidět na následujícím obrázku.

**HLAVNÍ MENU**

A screenshot of a computer

AI-generated content may be incorrect.

Toto okno představuje hlavní menu, ve kterém jsou tři ovládací prvky:

* Tlačítko **START** – spustí hru a odhalí okno herního módu.
* Tlačítko **LOAD** – otevře průzkumník souborů, kde uživatel může vybrat libovolný soubor JSON, který reprezentuje hrací level. Pokud navíc uživatel vybere takový soubor, že v dané složce jsou další JSON soubory, načte program do paměti všechny soubory v této složce a spouští při výhře rovnou další level.
* Tlačítko **MAP EDITOR** – odhalí okno editoru, kde uživatel může vytvářet vlastní levely.

**NAČTENÍ LEVELU**

Po stisku tlačítka LOAD se otevře průzkumník souborů, kde si uživatel může vybrat libovolný JSON soubor.

A screenshot of a computer

AI-generated content may be incorrect.

Tento soubor se pak stiskem „Otevřít“ zobrazí na hlavní obrazovce.

A screenshot of a video game

AI-generated content may be incorrect.

**HERNÍ MÓD**

Stisk tlačítka START odhalí okno herního módu, které lze vidě na následujícím obrázku.

A screenshot of a video game

AI-generated content may be incorrect.

V okně herního módu jsou tři ovládací prvky:

* Tlačítko **RESTART** – vrátí hru do původního stavu a uživatel ji může hrát odznovu.
* Tlačítko **PAUSE/RESUME** – stiskem tohoto tlačítka se hra pozastaví, opětovným stiskem se pak znovu spustí.
* Tlačítko **QUIT** – zastaví hru a vrátí uživatele do hlavního menu.

Pokud se Pacman střetne s monstrem nebo super monstrem, vyskočí na obrazovce hláška o prohře společně s dosavadním získaným skóre hráče.

A screenshot of a video game

AI-generated content may be incorrect.

Pokud se Pacmanovi podaří sníst všechny sušenky a vyhrát, vyskočí na obrazovce hláška o výhře společně s celkovým získaným skóre hráče.

A screenshot of a video game

AI-generated content may be incorrect.

**EDITOR MAP (LEVELŮ)**

Stiskem tlačítka MAP EDITOR v hlavním menu se uživatel dostane do editoru, kde lze upravovat existující a vytvářet nové mapy (levely). Okno editoru lze vidět na následujícím obrázku.

A screen shot of a computer

AI-generated content may be incorrect.

Editor obsahuje mnoho ovládacích prvků:

* **Kombinované pole „Select entity“** – po kliknutí se rozbalí seznam entit, které lze vložit do mapy. Po vybrání entity se v textovém poli pod nadpisem „Entity image path:“ zobrazí cesta k obrázku dané entity. Uživatel zde může zadat i vlastní cestu, pokud chce použít vlastní obrázek. Přidání entity do mapy lze provést kliknutím levým tlačítkem myši na libovolné místo v mapě. Odstranění entity lze provést stiskem pravého tlačítka myši na danou entitu.
* **TextBox TileSize a tlačítko GO** – do textového pole lze vyplnit velikost (hrany) políčka, tedy určit jak mají být vytvořené objekty velké. V původním nastavení má políčko rozměry 16x16, ale zadáním jiného čísla a potvrzením tlačítkem GO lze velikost přenastavit. Při přenastavení velikosti se dosavadní mapa vymaže a přenastaví rozměry pro novou mapu.
* **Tlačítko LOAD** – v textovém poli pod nadpisem „Background image path:“ lze zadat cestu k alternativnímu pozadí obrazovky. Po vyplnění platné cesty a stisknutí tlačítka LOAD se nové pozadí nastaví na hlavní obrazovce.
* **Tlačítko DEFAULT** – obnoví původní pozadí obrazovky.
* **Tlačítko CLEAR** – smaže všechny entity z mapy.
* **Tlačítko QUIT** – ukončí editor a vrátí uživatele zpět do hlavního menu.
* **Tlačítko SAVE** – uloží vytvořenou mapu do souboru JSON. Po stisknutí se otevře průzkumník souborů a jako výchozí jméno se nabídne mapX (kde X odpovídá dalšímu volnému číslu podle již existujících souborů).

A screenshot of a computer

AI-generated content may be incorrect.

**PROGRAMÁTORSKÁ DOKUMENTACE**

V této sekci se budu snažit zaměřit na popis implementace jednotlivých logických celků kódu. Detailní dokumentaci pak lze nalézt přímo v kódu ve formě komentářů.

**NÁSTIN ARCHITEKTURY**

Program je rozdělen do několika základních logických částí:

* **GameManager** – herní jádro, které řídí průběh hry a herní smyčku a zpracovává vstupy od uživatele.
* **Entity** a jejich potomci – logické objekty hry – Pacman, monstra, super monstra, sušenky, super sušenky a zdi.
* **GameObject** – vizuální reprezentace entity (sprite + souřadnice v mapě).
* **ScreenElement** – grafický prvek (obrázek nebo animace).
* **Map** – statická mapa, která obsahuje pozadí a rozmístění entit.
* **Renderer** – vykreslování pozadí a herních objektů.

Nyní rozebreru tyto logické části podrobněji.

**GAME MANAGER**

GameManager je hlavní řídící třída, mezi jejíž hlavní funkce patří, že:

* Obsahuje **herní smyčku** (*Update*).
* Uchovává herní stav pomocí vlastností: *isGameOff, isGamePaused, isGameWon, isSuperMode, CookiesCount* a *EatenCookies.*
* Spravuje 2D pole GameObjectů (*objects*), které představuje dynamickou herní mapu.
* Uchovává statickou instanci GameManageru ***instance***, ke které mají přístup všechny třídy a tak se propojuje celá logika hry.

**Metody:**

1. ***Update()***

* Zpracovává vstupy od uživatele (stisknuté klávesy).
* Skrze delegáty (v listu *onUpdateListeners*) volá metody *Update()* u všech objektů, které implementují interface *IUpdatable* a tedy se nějakým způsobem pohybují. To se týká Pacmana, monster a super monster.
* Odstraňuje delegáty metod u entit, které byly zničeny (tyto delegáti jsou přidáváni do listu *toBeRemovedListeners* při řešení kolizí).
* Spravuje zamaskované entity (když monster nebo super monster překryjí sušenku).

1. ***UpdateAnimations()***

* Pro všechny GameObjects volá metodu *UpdateAnimation()* a pokud je jejich sprite složen z více než jednoho obrázku, nechá vykreslit další obrázek v pořadí, tak se tvoří animace.

1. ***GetPressedKeys()***

* Registruje uživatelem stisknuté klávesy a přidává je do fronty *actionKeys*. Tato metoda zajišuje, že se za jeden cyklus *Update()* registruje pouze jeden stisk.

1. ***Reset()***

* Resetuje herní stav zpět na začátek, tedy vyprázdní listy *onUpdateListeners*, *toBeRemovedListeners*, *maskedObjects* a frontu *actionKeys* a vytvoří nové 2D pole GameObjects.

1. ***SuccesfullyEndGame(int score)***

* V případě dokončení levelu (pokud se *EatenCookies* rovná *CookiesCount*) zastaví hru a vypíše hlášku o výhře spolu se získaným skóre.

1. ***ClearMap()***

* „Vyčistí“ dynamickou mapu *objects* tak, že všechny políčka mapy nastaví na null.

**GAME OBJECT**

* GameObject propojuje *Entity* a její vizuální reprezentaci *ScreenElement*.
* Při vytvoření dostává sprite a počáteční souřadnice entity, aby bylo možné potom s objektem v mapě hýbat.
* Obsahuje metodu ***SetUpdateDelegate(UpdateDelegate updateDelegate)***, která umožňuje registraci delegáta do listu *onUpdateListeners*. GameManager pak volá *Update()* objektu při každém updatu.

**SCREEN ELEMENT**

* ScreenElement představuje grafickou reprezentaci entity.
* Ukládá šířku (*Width*) a výšku (*Height*) obrázku společně s cestou (*PathToSource*) k obrázku/složce s obrázky pro animaci a BitmapImage *ImageSource*, který představuje obrázek pro rendrování.

**Metody**:

1. ***LoadImage()***

* Zkontroluje jestli je cesta platná a pokud ano, tak zda je to cesta k obrázku, nebo složce.
* Pokud cesta není platná, vypíše hlášku „Path does not exist.“ a ukončí se.
* Pokud cesta vede k obrázku, vyvtoří a nastaví *BitmapImage*, který uloží do *imageSource*, který představuje obrázek pro rendrování.
* Pokud cesta vede ke složce, načte všechny obrázky ze této složky jako *BitmapImage* a uloží je do listu *animation*.

1. ***UpdateAnimation()***

* Pracuje s listem *animation* a počítadlem *frameCounter*.
* Při každém updatu vybere z listu *animation* nadcházející obrázek pro renderování, který uloží do *imageSource*. Pokud se dostane na poslední obrázek, *frameCounter* se vynuluje a posune se tak zpět na začátek (první obrázek).

**RENDERER**

* Renderer vykresluje hru do *Image*.
* **Metoda *Render(Map map, GameObject[,] objects)*:**
* Dostane jako argument instanci *Map*, ve které je uložená statická mapa, včetně rozměrů mapy, rozměrů jednotlivých políček mapy a cestě k obrázku na pozadí. Druhým argumentem je pak dynamické pole *objects*, ve kterém jsou uložené všechny game objecty.
* Metoda používá pro vykreslování třídy *DrawingVisual*, *DrawingContext* a *RenderTargetBitmap*, která z *Visual* sestaví *BitmapSource*.
* Nejprve vykreslí pozadí mapy.
* Poté projde všechny objekty v poli *objects* a vykreslí jejich sprite na správné souřadnice.
* Pokud je Pacman v hunter (super) módu, překryje obrazovku šedým poloprůhledným filtrem.

**MAP**

* Map reprezentuje statickou herní mapu, tedy mapu, která je před začátkem hry načtena.
* Uchovává rozměry mapy (*Width*, *Height*), rozměry jednotlivých políček mapy (*TileSizeWidth*, *TileSizeHeight*), cestu k pozadí (*PathToBackground*), screen element reprezentující obrázek pozadí (*Background*), cestu k mapě, kterou si ukládáme z důvodu nalezení všech map ve stejné složce (*PathToMap*) a 2D pole *Entity[,] map*, které předtavuje statickou mapu entit.
* Při vytvoření instance Map se nastaví *Width*, *Height*, *TileSizeWidth* a *TileSizeHeight*, načte se defaultní pozadí a inicializuje se pole *map*.

**Metody:**

1. ***LoadMap()***

* Inicializuje dvourozměrné pole entit *map* podle zadaných rozměrů tak, že vydělí celkovou výšku obrazovky výškou jednoho pole a celkovou šířku obrazovky šířkou jednoho pole.

1. ***ChangeBackgroundSource(string userPath)***

* Nastaví na pozadí alternativní obrázek tak, že přepíše *PathToBackground* na *userPath*, kterou metoda dostane jako argument, vytvoří nový screen element *background* a zavolá funkci *LoadImage()*. Pokud se obrázek podaří načíst, přepíše *Backgoround* na *background*.

1. ***SaveUserMap()***

* Serializuje instanci *Map* do JSON pomocí tříd *JsonSerializerSettings* a *JsonConverter*, které jsou dostupné v Nuggetovém balíčku *Newtonsoft.Json*.
* Po serializaci otevře průzkumník souborů a uloží soubor na uživatelem zvolené místo.

1. ***GetName()***

* Prohledá soubory v aktuální složce a vrátí string „mapX“ kde X odpovídá dalšímu volnému číslu podle již existujících souborů.

1. ***LoadUserMap()***

* Otevře průzkumník souborů, kde uživatel zvolí JSON soubor, který chce načíst.
* Deserializuje vybraný soubor pokud je validní, pokud ne, uloží do *PathToMap* prázdný řetězec a vratí null.

1. ***LoadUserMap(string path)***

* Používá se při načítání dalšího souboru z jedné složky. Tedy pokud dříve vybraná mapa byla ve složce s více dalšími mapami, tak se načetly všechny do jedné fronty a při splnění jedné mapy se automaticky načte další pomocí této metody.

**IUPDATEABLE**

* IUpdatable je interface, který vynucuje třídám, které ho implementují, implementovat metodu *Update()*. To se týká třídy Pacman, Monster a SuperMonster.

**ENTITY**

* Entity je abstraktní třída představující logický objekt hry. Třídu Entity dědí všechny objekty mapy a tedy všechny mají základní společné vlastnosti.
* Každá entita má souřadnice (*Coordinates*) X a Y, které určují pozici entity v mapě, cestu k obrázku, který jí náleží (*ImagePath*) a cross referenci na *GameObject* *gameObject* pro vytvoření screen elementu (renderování).
* Obsahuje dvě abstraktní metody: ***Destroy()*** pro zničení entity při kolizích a ***CreateGameObject(ScreenElement screenElement)*** pro vytvoření vizuální reprezentace entity.

**PACMAN**

* Tato třída reprezentuje hráče (Pacmana), dědí od třídy *Entity* a implementuje interface *IUpdatable* (implementuje metodu *Update()*).
* Uchovává získané skóre (*Score*), dobu trvání hunter módu (*HunterModeDuration*), pomocnou proměnnou pro sledování trvání hunter módu (*hunterModeTimeLeft*) a referenci na globální statickou instanci *GameManageru*.

**Metody:**

1. ***Destroy()***

* Zničí postavu pacmana, tedy vyřadí jeho delegáta na metodu *Update()* z listu *onUpdateListeners*, zastaví hru a vypíše hlášku o prohře společně s dosavadním získaným skóre.

1. ***Update()***

* Aktualizuje stav pacmana při každém updatu GameManageru.
* Kontroluje zda je Pacman v hunter módu (*isSuperMode*) a délku hunter módu. Při dosažení doby módu pak mód vypíná.
* Zpracovává vstupy od uživatele: uloží si aktuální souřadnice pacmana (*actualCoordinateX*, *actualCoordinateY*), inicializuje *deltaX* a *deltaY*, uloží si rozměry dynamické mapy (*sizeRow*, *sizeColumn*) a z *actionKeys* vytáhne z fronty stiknutou klávesu.
* Podle klávesy se pak *deltaX* nebo *deltaY* změní na +1 nebo -1 („A“ znamená doleva, tedy delta X se nastaví na -1).
* Zkontroluje jestli zadaný pohyb nepřekročí hranice mapy.
* Zkontroluje jestli na políčku, kam se chce Pacman posunout, je null, nebo se tam nachází nějaký jiný game object. Pokud je tam null, pacman se rovnou posouvá na dané políčko.
* Pokud políčko není prázdné a nachází se na něm jiný game object, musí se vyřešit kolize s tímto objektem. Po vyřešení kolize se v záležitosti na situaci pacman buď posune, nebo zůstane namístě, nebo prohrává.
* Kontroluje jestli jsou sežrány všechny sušenky porovnáním *CookiesCount* a *EatenCookies*, a v případě rovnosti zavolá metodu *SuccesfullyEndGame(Score).*

1. ***HandleCollision(Entity entity)***

* Řeší kolize pacmana s jinými game objecty. Metoda vrací bool, pokud vrátí true, je to znamení, že se pacman může posunout. Pokud vrátí false, pacman se neposouvá.
* Zde je změna oproti specifikaci: ve specifikaci je navrženo řešit kolize pomocí generických metod, avšak díky návrhu abstraktní třídy Entity od které všechny game objecty dědí byl problém vyřešen tímto způsobem (switch podle entity type).
* **Kolize s monstrem**: pokud je pacman v hunter módu (*isSuperMode*), zničí monstrum (na monstru se zavolá metoda *Destroy()*) a vrátí true, pokud není v hunter módu, zničí pacmana.
* **Kolize se super monstrem**: zničí pacmana (zavolá na pacmanovi metodu *Destroy()*) jelikož super monstra jsou nezničitelná.
* **Kolize se sušenkou**: ke skóre pacmana (*Score*) se přičte skóre sušenky, přičte se +1 k *EatenCookies*, na sušence se zavolá metoda *Destroy()* a vrátí true.
* **Kolize se super sušenkou**: zapne hunter mód (*isSuperMode*), ke skóre pacmana (*Score*) se přičte skóre super sušenky, přičte se +1 k *EatenCookies*, na sušence se zavolá metoda *Destroy()* a vrátí true.
* **Kolize se zdí**: vrátí false.

**MONSTER**

* Tato třída reprezentuje běžné monstrum, které se pohybuje po mapě náhodně a je schopné při kontaktu zničit pacmana, pokud není v hunter módu. Dědí od třídy *Entity* a implementuje interface *IUpdatable* (implementuje metodu *Update()*)
* Uchovává referenci na statickou instanci *GameManageru*, pevně nastavenou vlastnost *UpdateFreq*, která určuje frekvenci s jakou se na monstru provádí update (při stejné frekvenci jako pro pacmana se monstra pohybovaly až moc rychle) a pomocnou vlastnost *LastUpdate*, která uchovává naakumulovaný čas od posledního updatu.

**Metody:**

1. ***Destroy()***

* Zničí monstrum, tedy vyřadí jeho delegáta na metodu *Update()* z listu *onUpdateListeners* ana místu v poli, kde se nachází, přiřadí null.

1. ***Update()***

* Zkontroluje jestli monstrum není zničené (*isDestroyed*) a pokud ano, vrátí se a neprovádí update.
* Podle LastUpdate rozhodne jestli má provést update.
* Uloží si aktuální souřadnice monstra (*actualCoordinateX*, *actualCoordinateY*), inicializuje *deltaX* a *deltaY* a nastaví je na náhodnou hodnotu od -1 do 1 a uloží si rozměry dynamické mapy (*sizeRow*, *sizeColumn*).
* Zkontroluje jestli zadaný pohyb nepřekročí hranice mapy.
* Zkontroluje jestli na políčku, kam se chce monstrum posunout, je null, nebo se tam nachází nějaký jiný game object. Pokud je tam null, monstrum se rovnou posouvá na dané políčko.
* Pokud políčko není prázdné a nachází se na něm jiný game object, musí se vyřešit kolize s tímto objektem. Po vyřešení kolize se v záležitosti na situaci monstrum buď posune, nebo zůstane na místě, nebo je zničeno.

1. ***HandleCollision(Entity entity)***

* Řeší kolize monstra s jinými game objecty. Metoda vrací bool. Pokud vrátí true, je to znamení, že se monstrum může posunout. Pokud vrátí false, monstrum se neposouvá.
* **Kolize s pacmanem**: pokud je pacman v hunter módu (*isSuperMode*), zničí monstrum (na monstru se zavolá metoda *Destroy()*) a vrátí true, pokud není v hunter módu, zničí pacmana.
* **Kolize s monstrem**: vrací false a neposouvá se (odrazí se od sebe)
* **Kolize se super monstrem**: vrací false a neposouvá se (odrazí se od sebe)
* **Kolize se sušenkou**: přidá sušenku do listu zamaskovaných objektů (*maskedObjects*) a překryje ji, tedy vrátí true.
* **Kolize se super sušenkou**: přidá super sušenku do listu zamaskovaných objektů (*maskedObjects*) a překryje ji, tedy vrátí true.
* **Kolize se zdí**: vrátí false.

**SUPER MONSTER**

* Tato třída reprezentuje super monstrum, které se pohybuje po mapě náhodně a při kontatku s pacmanem ho vždy zničí. Dědí od třídy *Entity* a implementuje interface *IUpdatable* (implementuje metodu *Update()*)
* Od třídy *Monster* se liší jen tím, že implementuje prázdnou metodu *Destroy()* jelikož super monstrum nelze zničit a při řešení kolizí vždy pacmana zničí.

**COOKIE A SUPERCOOKIE**

* Třídy reprezentující sušenky a super sušenky, dědí od třídy Entity.
* Uchovávají své fixně zvolené skóre (*Score* – pro Cookie 100, pro SuperCookie 500) a referenci na statickou instanci *GameManageru*.
* Implementují metodu *Destroy()* která je označí jako zničené (*isDestroyed*) a vymaže je z dynamické mapy (*objects*).

**WALL**

* Třída reprezentující zeď, dědí od třídy Entity.
* Není pohyblivá, ani nemůže být zničena a tak implementuje pouze prázdnou metodu *Destroy()* a metodu *CreateGameObject().*

**MAINWINDOW**

* Třída MainWindow.xaml.cs řídí hlavní okno WPF aplikace, spravuje vykreslování hry pomocí instance třídy Renderer, řídí herní smyčku pomocí instance GameManageru, obsahuje inicializaci hry a všechny ovládací prvky.
* Uchovává statickou mapu *Map*, *renderer*, který je zodpovědný za vykreslování, *GameManager gameManager*, *Stopwatch gameUpdateWatch* pro kontrolu intervalu updatování hry, *Stopwatch animationWatch* pro kontrolu intervalu updatování animací a frontu *pathToMaps*, do které se uloží všechny JSON soubory z jedné vybrané složky (při načtení jedné mapy se načtou do fronty všechny ostaní JSON soubory z dané složky a po splnění se spouští automaticky další)
* Detailní dokumentace ke každé metodě je dostupná v kódu ve formě komentářů, nicméně dokumentace některých metod, které nejsou jen event handlery, následuje níže.

**Metody:**

1. ***UpdateUI(object sender, EventArgs e)***

* Metoda *UpdateUI* je při inicializaci programu přiřazena do rendering eventu WPF aplikace (*CompositionTarget.Rendering*).
* Kontroluje jestli je hra vyhrána (dokončen level) a pokud ano, zkontroluje zda je fronta *pathToMaps* prázdná. Pokud ano, vypíše hlášku „All levels won!“. Pokud ne, načte další mapu (level) z fronty *pathToMaps*.
* Pokud hra neběží (*isGameOff*), resetuje dynamické objekty mapy, aby změny v mapě (v map editoru) byly ihned viditelné. Tedy pro všechny políčka v mapě zkopíruje všechny objekty ze statické mapy (*Map.map*) do dynamické mapy (*objects*).
* Skrze renderer spouští metodu *Render(Map map, GameObject[,] objects)* a vykresluje mapu a game objecty.
* Pomocí *Stopwatch gameUpdateWatch* a pevně daného času (350000 milisekund) kontroluje zda zavolat *Update() GameManageru*.
* Pomocí *Stopwatch animationWatch* a pevně daného času (2000000 milisekund) kontroluje zda aktualizovat animace.

1. ***OverlayCanvas\_InsertObjectHandler(object sender, MouseButtonEventArgs e)***

* Určí přesné místo na hlavní obrazovce, kde uživatel stiskl pravé tlačítko myši.
* Podle souřadnic kliknutí určí pozici v poli (*row* a *column*)
* Podle entity vybrané uživatelem v *ComboBoxu* se vytvoří na daném místě nový game object, který se přiřadí do statické i dynamické mapy.

1. ***OverlayCanvas\_DeleteObjectHandler(object sender, MouseButtonEventArgs e)***

* Stejně jako předchozí metoda určí místo stisku levého tlačítka myši a z tohoto místa odstraní entitu, pokud tam nějaká existuje.

1. ***LoadJsonLevels()***

* Pomocí *PathToMap* nalezne cestu ke složce aktuálního souboru.
* Zkontroluje, že cesta je validní a uloží cestu ke složce.
* Načte ze složky všechny další existující JSON soubory a uloží je do fronty *pathToMaps*, přičemž z fronty vyřadí aktuální mapu.

1. ***LoadMap(Map map)***

* Vyčistí všechny dynamické objekty ze stávající dynamické mapy (*ClearMap()*) a resetuje herní stav (*Reset()*).
* Načte mapu, kterou dostane jako argument. Tedy načte pozadí (*ChangeBackgroundSource()*) a všechny herní objekty (nejdříve vytvoří ScreenElement na kterém zavolá metodu *LoadImage()* a poté vytvoří game object zavoláním metody *CreateGameObject()*).