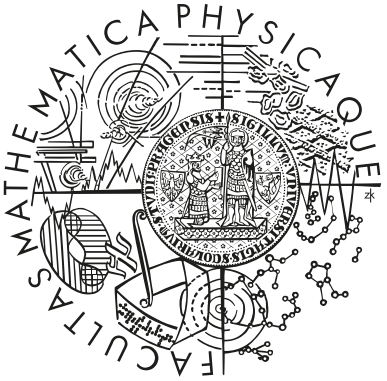
Univerzita Karlova

Matematicko-fyzikální fakulta

Studijní program: Informatika



Programování II
Dokumentace k zápočtovému programu
Bomberman: CPU Wars

2022/2023 **Jiří Daněk**

Anotace:

Přetvoření známé hry Bomberman s herními prvky z oboru IT. Jedná se o lokální multiplayer pro 2-4 hráče napsán v jazyku C# ve vývojovém herním enginu Unity. Hráči si mohou vybrat ze tří různých map a mají za úkol pokládat vybuchující CPU a tím zničit nejen počítače, ze kterých padají vylepšení, ale především eliminovat ostatní hráče. Poslední přeživší obdrží jeden bod, hru vyhrává hráč s nejvyšším skóre.

Annotation:

Recreation of the well-known Bomberman game with gameplay elements from IT. It is a local multiplayer for 2-4 players written in C# using the Unity game engine. Players can choose from three different maps and are tasked with placing exploding CPUs to destroy not only the computers from which the upgrades fall, but most importantly to eliminate other players. The last survivor receives one point, the game is won by the player with the highest score.

1. Zadání

Přepracování hry Bomberman ve vývojovém herním enginu Unity za pomocí jazyka C#. Ve hře jsou implementovány různé mapy, které si hráči mohou volit. Možnost úpravy počátku hry v nastavení, kde si hráč vybere počáteční množství bomb, velikost výbuchu, rychlost postav, pravděpodobnost výskytu vylepšení po rozbití bedny a úprava hlasitosti hudby na pozadí. Hráčí pokládají výbušné objekty, kterými ničí rozbitné boxy a eliminují ostatní hráče, poslední hráč získá bod do své tabullky skóre, která je udržována po dobu hraní stejného hracího módu.

2. Uživatelská část dokumentace

I. Úvod

Vítejte v uživatelské dokumentaci hry Bomberman, lokalizovaný multiplayerový titul, který vás zavede do světa plného akce a strategie. Tento dokument vám poskytne detailní informace o ovládání, možnostech nastavení a dalších důležitých prvcích hry.

II. Popis hry

Hra Bomberman se odehrává na vybrané mapě, kde se 2-4 hráči pohybují a umisťují CPU, které ničí protihráče a boxy reprezenotvané počítači. Po určité době dojde k výbuchu CPU a objeví se binární kód – výbuch. Při zničení počítače se se zvolenou pravděpodobností objeví náhodné ze tří vylepšení. Vaším cílem je eliminovat protihráče a získat bod za každé přežité kolo. Hráč s nejvyšším počtem bodů se stává vítězem. Hra nabízí 3 různé mapy a 4 postavy.

III. Ovládání (pohyb, bomba)

i. HERNÍ PROSTŘEDÍ – klikání kurzorem na tlačítka
ii. ZPĚTNÉ TLAČÍTKO – ESC
iii. HRÁČ 1 – WASD, E
iv. HRÁČ 2 – ŠIPKY, ENTER
v. HRÁČ 3- IJKL, O
vi. HRÁČ 4 – TFGH, Z

IV. Orientace v herním prostředí

Po spuštění hrz se objevíte v hlavním menu, zde je na výběr QUIT – opustí hru a vymaže momentální nastavení hry, SETTINGS – nastavení herní konfigurace např. počáteční rychlost postavy, počet bomb, pravděpodobnost výskytu vylepšení, hlasitost …, GAME MODES konfigurace herního režimu, přesun do další scény s možností výběru mapy.

Stisknutím tlačítka GAME MODES se přesuneme na výběr mapy: MAP1, MAP2, MAP3. Při najetí na tlačítko se ukáže vizualizace mapy, mimojiné je zde i tlačítko RETURN, které vrací uživatele zpět do hlavního menu.

Výběrem mapy se dostaneme na scénu s možností výběru počtu hráčů: 2 PLAYERS, 3 PLAYERS, 4 PLAYERS. Dle vybrané konfigurace se načte uživatelem vzbrané počíteční nastavení hráčů a dle herní konfigurace se zobrazí mapa a počet hráčů a jejich počáteční skóre 0, nebo X reprezentující nezúčastěného hráče. Mimojiné se objeví okno s nápisem PRESS ESCAPE TO RETURN TO MAIN MENU, neboli upozornění, že po stisku tlačítka ESC se uživatel dostane zpět do hlavního menu a anuluje se herní skóre, volba mapy a počtu hráčů. Tímto okamžikem může hra naplno odstartovat.

V. Uživatelský závěr

Věřím, že vám tato dokumentace poskytla veškeré potřebné informace ke skvělému zážitku při hraní Bombermana. Nyní jste připraveni vstoupit do světa CPU Wars a zažít napínavé souboje a strategické rozhodování. Děkuji a přeji vám příjemné hraní!

3. Programátorská část dokumentace

I. Scény

Celé uživatelské rozhraní je rozděleno do 6 scén. Main menu, Settings menu, Map menu a výběr tří odlišných map. Scény s názvem menu jsou ovládány pomocí kurzoru a levého tlačítka značícího výběr. Scény s mapami již simulují hru a zde hra reaguje pouze na výše specifikované hrací klávesy a Escape. Font použitý jak pro reprezentaci skóre tak i pro jiné jmenovky se nazývá Press Start 2P. Herní grafika je vytvořena přímo pro tuto hru v programu PixilArt.

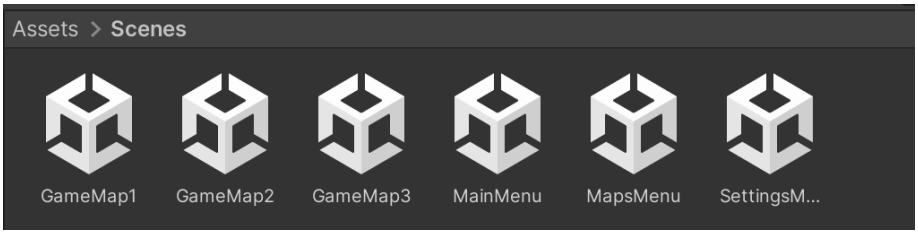


Illustration 1: Vytvořené scény v prostředí Unity

i. Main menu

První obrazovkou je hlavní menu, která se zobrazí ihned po spuštění hry. Na pozadí se nachází tématický herní obrázek. Jediným ovládacím prvkem této stránky jsou 3 tlačítka. GAME MODES – přejde na scénu Map menu, SETTINGS – přejde na scénu Settings Menu, QUIT – vymaže uložené PlayerPrefs a ukončí aplikaci. V poslední řadě se zde nachází hudba hrající v pozadí.

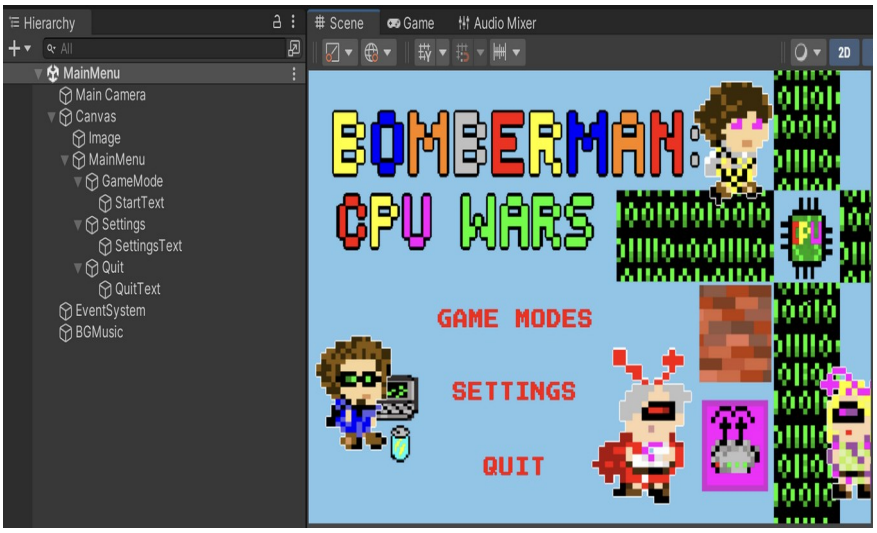


Illustration 2: Scéna MainMenu s jejími objekty

ii. Settings menu

Druhou obrazovkou je Settings menu, kde se nachází jak nastavení hry tak i nastavení zvuku a displeje – funguje pouze pro aplikaci, nikoliv WebGL variantu. Zde se vyskytuje řada UI prvků, kterými jsou slider, regulující hlasitost hudby na pozadí, sada dropdown objektů popisující počáteční nastavení hráčů – počáteční množství bomb, počáteční, pravděpodobnost výskytu vylepšení při rozbití boxu, počáteční velikost výbuchu a také rozlišení obrazovky. Posledním objektem v této scéně je tlačítko RETURN pro návrat do Main manu.

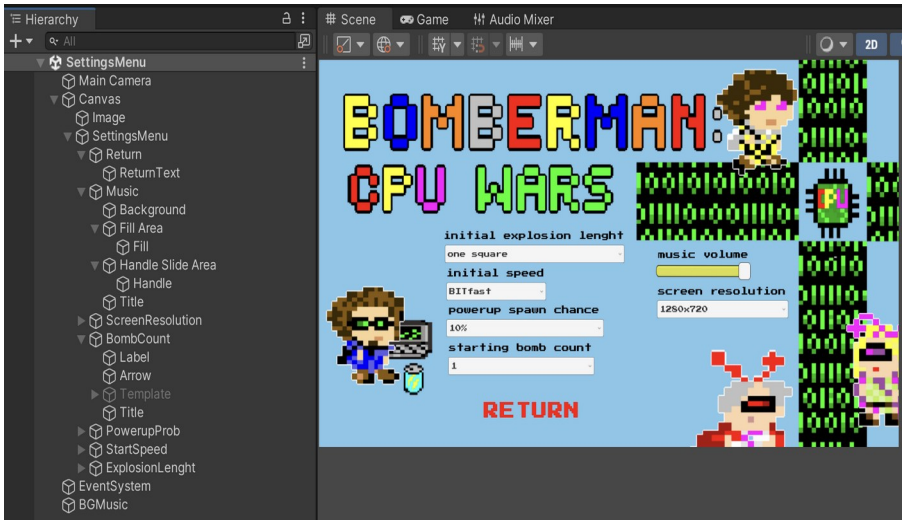


Illustration 3: Scéna SettingsMenu s jejími objekty

iii. Map menu

Třetí obrazovkou se kterou se uživatel setká je Map menu. Zde se pouze změní tlačítka a jejich akce. Vyskytují se zde tlačítka MAP1, MAP2, MAP3 – reagují jak na kliknutí, jenž přesune uživatele na zvolenou scénu obsahující příslušnou mapu tak i na najetí kurzorem na tlačítko, čímž se automaticky zobrazí malé okénko ilustrující mapu. V poslední řade se zde vyskytuje tlačítko RETURN, které vrací uživatele zpět do Main menu.

Illustration 4: Scéna MapsMenu s jejími objekty

iv. Konkrétní mapy

Poslední scénou jsou konkrétní mapy – GameMap?, které se od sebe liší pouze objektem Grid, tedy rozložením rozbitných boxů – počítače, nerozbitných boxů – cihlové stěny, podlahy a rozmístěním hráčů. První interakcí na této scéně jsou tlačítka pro výběr počtu hráčů z objektu PreGameScene. 2 PLAYERS, 3 PLAYERS, 4 PLAYERS. Dalším již je pouze tlačítko RETURN pro návrat do Main menu. Po tomto výběru se objekt deaktivuje a již se zobrazí vybraná konfigurace mapy, počet hráčů, inicializované jednotlivé skóre na hodnotu 0. Na několik sekund se také zobrazí hláška oznamující návratovou klávesu do Main menu. Hlavním objektem na scéně jsou hráči (Player?), které obshaují všechny potřebné skripty s metodami. Mimo jiné je zde vytvořen i objekt Scoreboards, který se stará o skóre hráčů, viz script níže.

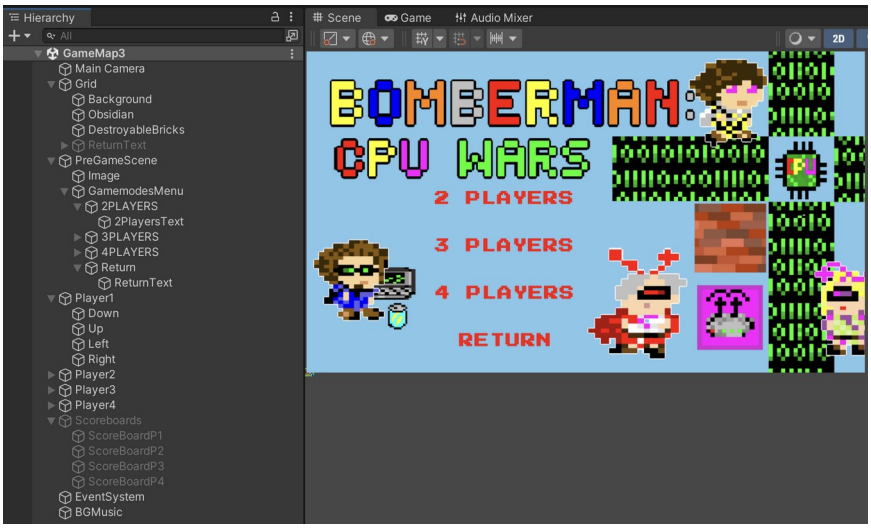


Illustration 5: Scéna GameMap? s jejími objekty

Illustration 6: Příklad jednoho z herních prostředí

II. Zvuky ve hře

Ve hře jsou implementovány pouze dva prvky audia. Prvním je hudba na pozadí od autora Dream-Protocol, která je royalty free a dalším je výbuch CPU pod názvem mixkit 8 bit bomb explosion.

III. Ovladaní hry

Hra se dělí na ty scény, které konfigurují danou hru a na scény se samotným multiplayerem. Konfigurační scény se ovládají výhradně pomocí kuzoru a klikáním levého potvrzovacího tlačítka. Scéna hry se ovládá pomocí interakce s definovanými klávesami v uživatelské části dokumentace. Ukládáni parametrů ve scéně Settings menu se automaticky uloží do PlayerPrefs bez nutnosti dalších činností.

IV. Objekty a jejich animace

Samotné prostředí se dělí na řadu objektů které jsou animovány pomocí vlastní vytvořené grafiky, viz GITLAB a UI v Unity prostředí. Grafika je importována do Unity Assets/PixelArt a upravena na 16 pixelů na jednotku, odebrán filter mód a formát nastaven na RGBA 32 bit.

Objekty tlačítek, dropdown a posuvník jsou nativně podporovány prostředím Unity v záložce UI, která nabízí i jejich triviální modifikaci, animaci a funkcionality. Tyto objekty reagují jak na kliknutí tak na najetí kurzorem. Ke každému tlačítku je vázána funkce ze skriptu MainMenu.cs, SettingsMenu.cs, GamemodeSelection.cs, viz níže.

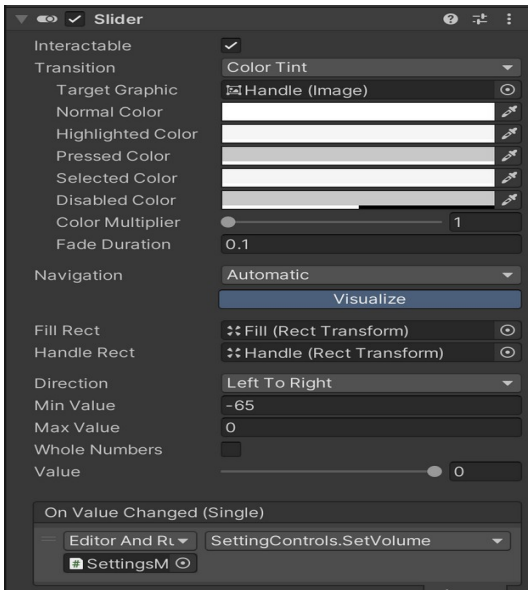


Illustration 7: Příklad posuvníku pro regulaci hlasitosti audia

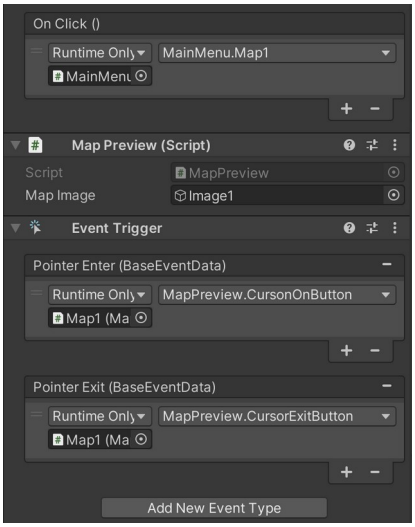


Illustration 8: Příklad tlačítka pro výběr mapy

Mapy i pozadí jsou vytvořeny pomocí nástroje Tilemap v Unity, přičmž byla vytvořena paleta obsahující jednotky z Git viz obrázek níže. Tilemap byla podrozdělena do tří vrstev se stejnými tagy.

i. Background – vrstva na představující převážně podlahu, u které byl nastaven Collider Type na hodnotu None a další prvky jako voda, tráva, cedule…

ii. Wall – vrstva představující stěnu, tedy ty prvky, které ohraničují území za které hráči nemohou vstoupit, Collider Type má hodnotu Sprite

iii. Destructibles – vrstva pro nastavení rozložení rozbitných boxů, ze kterých jsou generovány vylepšení. V tomto případě grafika počítačů, Collider Type nastaven na hodnotu Sprite.

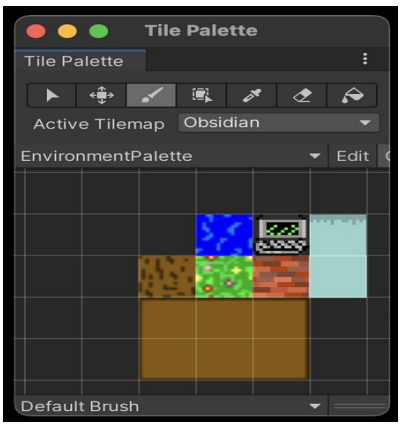


Illustration 9: Paleta pro ilustraci herní mapy

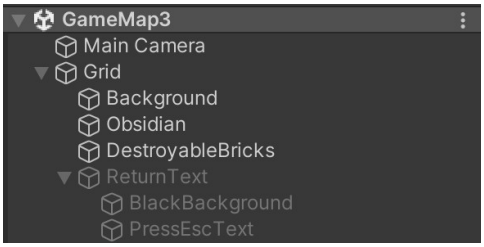


Illustration 10: Rozložení vrstev v objektu Grid

Jednotlivé **postavy** přijímají vstup z klávesnice a dle zvolené klávesy simulují pohyb daným směrem. Každá postava má podtřídy reprezentující daný pohyb, třída je aktivována právě tehkdy když dojde k aktivaci definované klávesy a zůstane aktivní dokud nedojde ke změně pohybu. K animaci hrdinů slouží skript DirectionAnimation.cs popsaný níže či k nahlédnutí v příloze Scripts v Git repozitáři. Tento skript má dva stavy idle/running, které se mění dle stisku kláves.

Illustration 12: Objekt hráče a jeho podobjekty

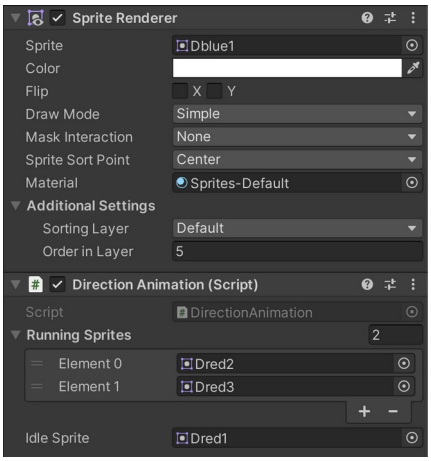


Illustration 11: Příklad animace jednoho směru hráče

Bomba (CPU) je instance Prefab, která je generována na místo hráče při splnění stiskutí aktivačního tlačítka a zároveň hráč má bomby k dispozici. Bomba obsahuje Collider a Rigidbody, která jsou ale aktivována až poté, co hráč opustí definovaný prostor, jinak by došlo k nežádoucí kolizi objektů. Zároveň se při položení bomby spustí Coroutine, která po definovaném čase zničí objekt a generuje Explozi. O pokládání bomb se stará skript BombPlace.cs, viz níže.

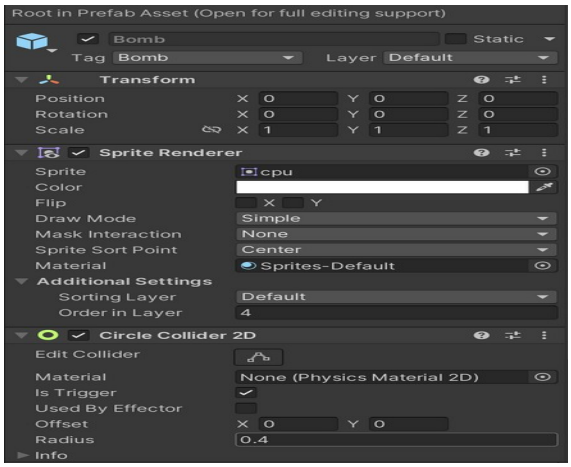


Illustration 13: Nastavení bomby v Prefabs

Exploze (binární kód) je instance Prefab, která je generována po konci Coroutine, kterou spouští položení bomby na hrací plochu. Exploze je komplikovaný objekt, který zajišťuje následující. Šíření exploze dle hráčova vylepšení, zastavení exploze o zeď (Wall) Tale, usmrcení hráče, likvidace ničitelných bloků. O tyto funkcionality se starají skripty BombExplosion.cs, DestructibleCollision.cs, Gameloop.cs ve spolupráci se skriptem BombPlant.cs z odstavce výše. Také generuje při vzniku zvukový efekt výbuchu.

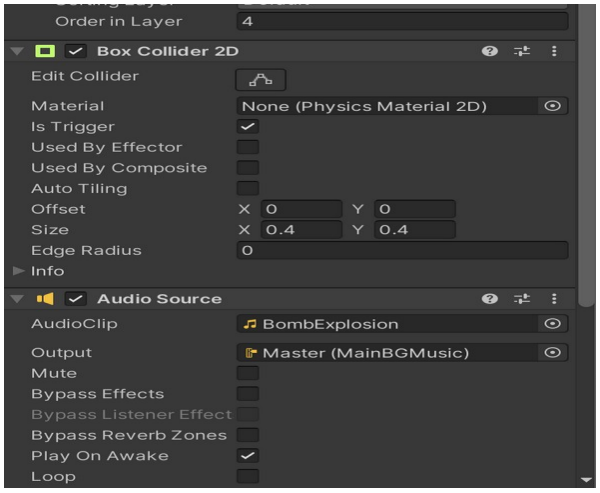


Illustration 14: Nastavení exploze v Prefabs

Scoreboards je třída vyvořená pomocí UI Text Mesh Pro starající se o uchování, zobrazovaní a anulování dosavadního skóre pokud je potřeba. Mimo jiné se díky nenávaznosti na objekty hráčů stará o kolize, kdy dojde k úmrtí dvou hráčů ve stejném snímku a restart hry – nové kolo.

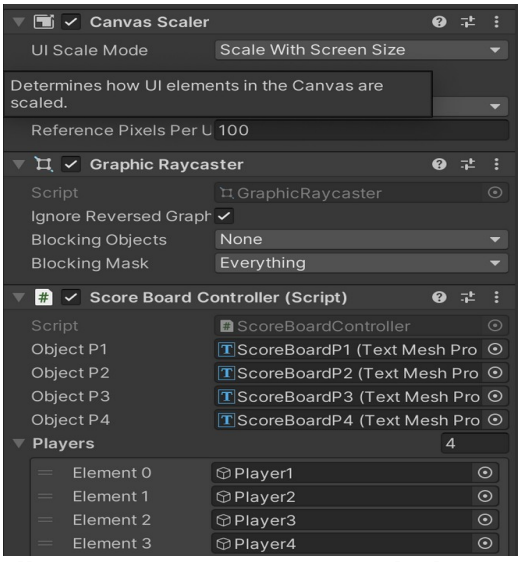


Illustration 15: Komponenty objektu Scoreboards

Vylepšení (ItemBlastRadius, ItemExtraCPU, ItemSpeedIncrease) jsou instance Prefabs, které jsou s nakonfigurovanou pravděpodobností generovány ze zničených boxů – počítačů. Jedná se o zvýšení dosahu výbuchu (router), zvýšení počtu bomb (CPU), zrychlení hráče (plechovka). O aktivaci vylepšení se stará skript DestructiblesCollisions.cs, který se stará jak o jejich generování, sbírání tak i o aktivaci.

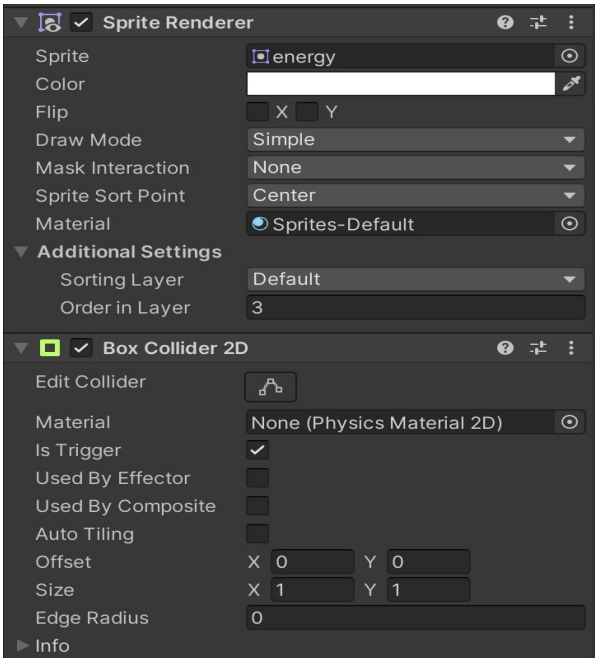


Illustration 16: Nastavení vylepšení Prefabs

V. Popis skriptů

Všechny skripty jsou psány v jazyce C# za pomocí knihoven Unity Unity.Engine, TMPro a základní System.

MainMenu.cs obsahuje různé metody, které se volají při interakci s tlačítky v hlavním menu hry – scéna MainMenu.

i. *Awake()* je volána při spuštění objektu a slouží k vymazání všech uložených dat. Pro každého z prvních čtyř hráčů je smazán klíč v PlayerPrefsech, který odpovídá jejich skóre. Také je smazán klíč "LastGamemode", který uchovává poslední vybraný herní mód.

ii. *MapsMenu()* slouží k přechodu na obrazovku výběru herního módu. Volá se při stisknutí tlačítka.

iii. *QuitGame()* slouží k ukončení hry. Nejprve jsou smazána všechna uložená data a poté je aplikace ukončena

iv. *ReturnToMain()* slouží k návratu do hlavního menu hry. Volá se při stisknutí tlačítka.

v. *Settings()* slouží k přechodu na obrazovku nastavení hry. Volá se při stisknutí tlačítka.

vi. Map?() slouží k přechodu na konkrétní herní mapu. Každá z metod volá příslušnou scénu s mapou.

MapPreview.cs skript slouží k zobrazení náhledu mapy při najetí myši na tlačítko. Proměnná MapImage je veřejná proměnná typu GameObject, která slouží k odkazu na herní objekt reprezentující mapový obrázek. Metody jsou přiřazeny ve scéně MapsMenu u jednotlivých tlačítek s pomocnou komponentou Event Trigger. Tento skript umožňuje zobrazit náhled mapy při najetí myši na tlačítko a skrýt ho při odjetí myši z tlačítka.

i. *CursorOnButton()* se volá při najetí myši na tlačítko a aktivuje mapový obrázek.

ii. *CursorExitButton()* se volá při odjetí myši z tlačítka a deaktivuje mapový obrázek.

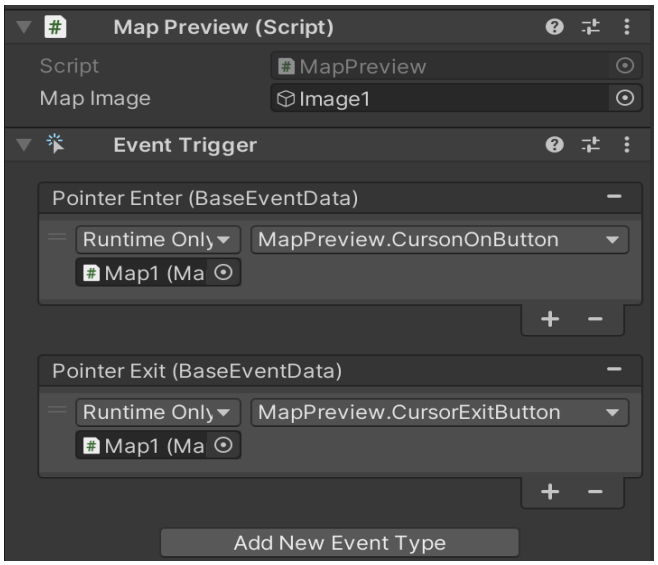


Illustration 17: Implemetnace MapPreview.cs s tlačítky

SettingControls.cs slouží k ovládání nastavení ve hře, jako je rozlišení obrazovky, hlasitost zvuku, počet bomb, pravděpodobnost power-upů atd. Veřejné Proměnné Resolution, BombCount, PowerupProb, StartSpeed a ExplosionLength jsou typu TMP_Dropdown z knihovny TextMeshPro, které slouží k odkazu na rozbalovací seznamy (dropdowny) v herním rozhraní. Proměnná MainMixer je odkaz na AudioMixer, který umožňuje nastavovat globální hlasitost zvuku ve hře. Proměnná AudioSlider je odkaz na posuvník, který slouží k ovládání hlasitosti zvuku. Konstantní proměnné jako storedResolution, storedBombCount, atd. slouží k ukládání hodnot nastavení do PlayerPrefs, což je mechanismus pro ukládání dat mezi herními sezeními. Tento skript umožňuje hráčům upravit různá nastavení ve hře, jako je rozlišení obrazovky, hlasitost zvuku a další. Ukládá tyto nastavení pomocí PlayerPrefs, aby se při příštím spuštění hry načetly poslední použité hodnoty.

i. *Awake()* je volána při inicializaci a přiřazuje posluchače událostí pro změnu hodnot rozbalovacích seznamů a posuvníku.

ii. *Start()* je volána na každém spuštění skriptu a nastavuje hodnoty objektů podle uložených hodnot v PlayerPrefs.

iii. Metody jako *ChangeExplosionLength()*, *ChangeStartSpeed()*, atd. jsou volány při změně hodnot rozbalovacích seznamů a ukládají nové hodnoty do PlayerPrefs.

iv. *SetVolume()* je volána při změně hodnoty posuvníku hlasitosti a nastavuje hlasitost zvuku pomocí AudioMixeru a ukládá novou hodnotu do PlayerPrefs

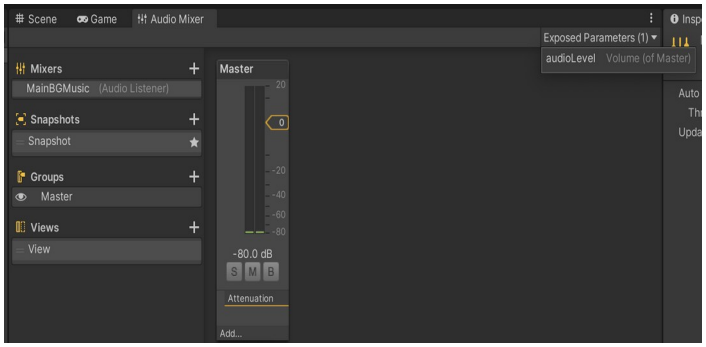


Illustration 18: Audio Mixer Controller

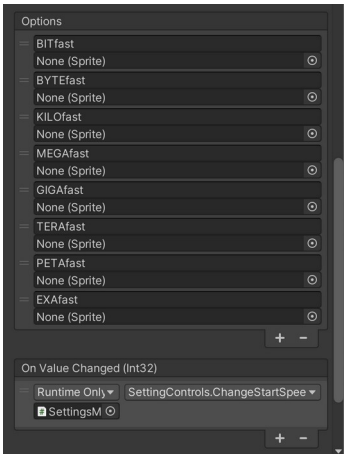


Illustration 19: Příklad nastavení dropdown menu

GameModeSelection.cs zajišťuje chování před spuštěním samotné hry. Jedná se o Objekt, který překrývá herní prostředí a vybírá se zde počet hráčů – nachází se tedy na scéně GameMap?. Proměnné Scoreboard a ReturnKeyText jsou reference na objekty ve scéně, které představují tabulku skóre a text s instrukcí pro návrat do hlavního menu. Konstantní proměnná storedLastGamemode slouží jako klíč pro uložení posledně vybraného herního režimu v PlayerPrefs.

i. *Awake()* se volá při inicializaci skriptu a zjišťuje, zda je hra již spuštěna. Pokud ano, na základě uloženého herního režimu v PlayerPrefs přepíná na odpovídající herní režim pomocí metod Multiplayer2, Multiplayer3 nebo Multiplayer4.

ii. *isGameRunning()* kontroluje, zda je hra již spuštěna, a to na základě součtu skóre uloženého v PlayerPrefs. Pokud je součet skóre větší než 0, znamená to, že byla alespoň jedna hra odehrána.

iii. *showScoreboards()* aktivuje objekt tabulky skóre ve scéně.

iv. *showQuitKeyText()* zobrazuje na počítku každé hry okno s instrukcemi pro návrat do hlavního menu po dobu 3 sekund. Okno je

reprezentováno objektem ReturnKeyText ve objektu Grid. Po uplynutí času je objekt deaktivován.

v. *hideGamemodesScene()* hledaá a deaktivuje objekt s výběrem herního režimu, objekt s tagem "Gamemode" a deaktivuje ho.

vi. *hideXPlayers()* deaktivuje dané hráče na základě vybraného herního režimu.

vii. *Multiplayer?()* jsou volány při vybrání příslušného herního režimu. Ukládají vybraný herní režim do PlayerPrefs, skrývají odpovídající počet hráčů, deaktivují scénu s výběrem herního režimu, aktivují tabulku skóre a zobrazují instrukce pro návrat do hlavního menu.

viii. *ReturnToMain()* slouží k přechodu na hlavní menu.

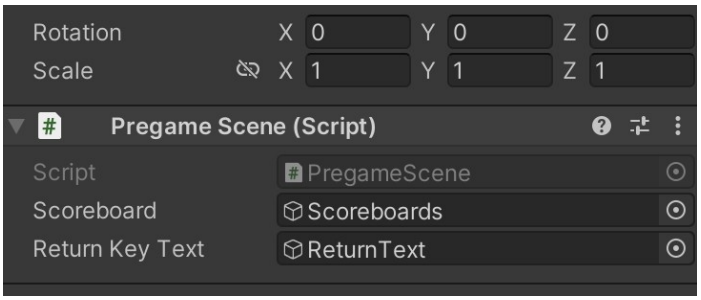


Illustration 20: Implementace GameModeSelection.cs

BombPlant.cs zajišťuje chování spojené s umisťováním a správou bomb. Skript je umístěn v objektu hráče. Dvě veřejné proměnné BombPrefab – představuje grafickou a herní reprezentaci bomby a BombDropKey – slouží k určení klávesy pro umístění bomby. Skript udržuje seznam objektů bomb v proměnné bombs, který představuje všechny umístěné bomby v herní scéně patřící danému hráči. Proměnné bombCount a bombLifetime určují počet bomb, které hráč může celkem umístit, a dobu, po kterou je bomba aktivní před výbuchem. Proměnná bombAvailable uchovává aktuální počet dostupných bomb, které hráč může momentálně umístit. Kromě toho skript uchovává data o hráčích, kteří jsou otagování jako Players, v poli players, které získá z herní scény pomocí metody FindGameObjectsWithTag(). Dále má proměnnou bombTriggerDistance, která určuje vzdálenost, ve které se bomba aktivuje jako objekt se kterým lze kolidovat – možnost bomby odsouvat. Skript také obsahuje odkaz na skript BombExplosion pomocí proměnné explosionScript, který řídí chování exploze bomby.

i. *Start()* získá počet bomb z metody getSetBombCount() a nastaví proměnnou bombAvailable na počet dostupných bomb. Poté se připraví odkaz na skript obsluhující exploze a uloží se reference na hráče.

ii. *GetSetBombCount()* zjistí zda-li byla upravena hodnota počtu bomb v nastavení hry a vrátí buď tuto hodnotu nebo implicitní 1.

iii. *Update()* reaguje na stisknutí klávesy BombDropKey a podmínkou ověřuje, zda má hráč k dispozici nějakou bombu. Pokud jsou tyto podmínky splněny, volá se metoda placeBomb().

iv. *steppedPlayerOutOfBomb()* kontroluje, zda se hráč vzdálil od bomby na dostatečnou vzdálenost, aby mohla bomba kolidovat s ostatními objekty ve scéně.

v. *placeBomb()* umisťuje bombu na aktuální zaokrouhlednou pozici hráče a následně vytváří nový objekt bomby pomocí Instantiate(), který je přidán do seznamu bombs pro další správu. Hráči snížen počet dostupných bomb a pokud hráč zemře dříve než bomba exploduje, je bomba zničena. Navíc se zde volá obsluha Coroutine, která se stará o explozi ve skriptu BombExplosions.cs.

vi. *IncrementBombCount()* inkrementuje bomb bombCount a bombAvailable, při získání power-upu.

vii. *IncrementBombAvailable()* inkrementuje bombAvailable a používá se pro obnovení počtu bomb po explozi.

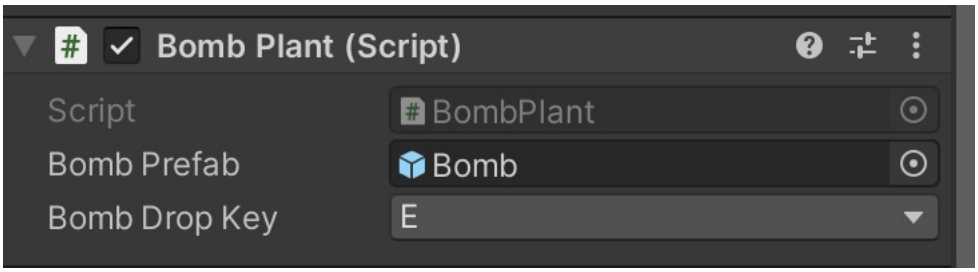


Illustration 21: Implementace BombPlant.cs

PlayerMovement.cs řídí pohyb hráče a je tedy v objektu hráče také umístěn. Proměnná rb představuje komponentu Rigidbody2D objektu hráče a proměnná direction určuje směr pohybu hráče na začátku hry a má výchozí hodnotu Vector2.down. Proměnná speed uchovává rychlost pohybu hráče. Klávesy pro ovládání pohybu jsou definovány pomocí proměnných UpKey, LeftKey, RightKey a DownKey. Skript také obsahuje animace pro jednotlivé směry pohybu hráče (DirUp, DirDown, DirLeft, DirRight) pomocí komponenty skriptu DirectionAnimation, který se nachází v podobjektech s danými směry objektu hráče.. Aktuálně vybraná animace/směr je uchováván v proměnné dirCurrent. Klávesa ReturnKey slouží k návratu do hlavního menu.

i. *Start()* získá počáteční rychlost pohybu z metody getSetStartSpeed(). Dále se nastaví aktuální animace na směr dolů a ostatní jsou vypnuty.

ii. *getSetStartSpeed()* získává počáteční rychlost pohybu hráče, kterou hráč může nastavit v před-herních volbách, jinak je rovna 2.

iii. *Awake()* je volána při inicializaci skriptu a získává komponentu Rigidbody2D pro pohyb hráče.

iv. *Update()* kontroluje stisknutí klávesy pro návrat do hlavního menu zjišťuje vstup od hráče pomocí metody checkInput() a provádí pohyb hráče pomocí metody movePlayer().

v. *checkIfReturnKey()* kontroluje, zda byla stisknuta klávesa pro návrat do hlavního menu. Pokud ano, načte se scéna MainMenu.

vi. *checkInput()* zjišťuje, které klávesy pro pohyb hráče byly stisknuty v daném snímku. Na základě stisknutých kláves se aktualizuje směr pohybu, animace a provádí se přepínání mezi animacemi směru.

vii. *upadtePosition()* aktualizuje animace a směr pohybu hráče na základě nového směru a animace. Vypíná a zapíná příslušné animace směru pohybu.

viii. *IncrementPlayerMovement()* inkrementuje rychlost pohybu. Používá se pro zvýšení rychlosti při získání power-upu.

ix. *movePlayer()* provádí samotný pohyb hráče na základě nové pozice vypočítané z aktuální pozice hráče, směru a rychlosti pohybu.

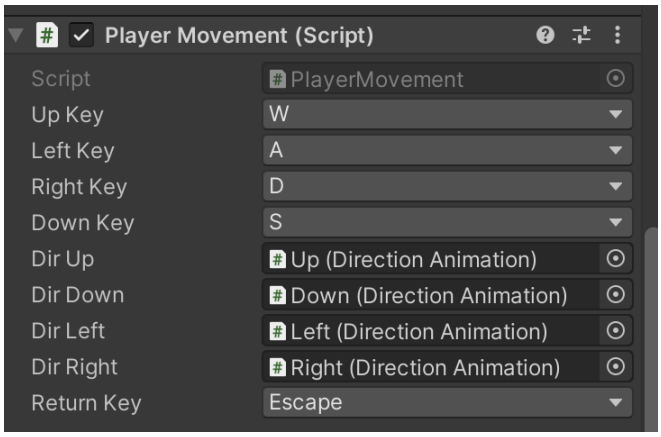


Illustration 22: Implementace PlayerMovement.cs

BombExplosion.cs řídí exploze bomby a je také umístěn v objektu hráče. Proměnná ExplosionPrefab je grafická/herní reprezentace exploze, zatímco LayerObsidian a LayerDestructible drží referenci na vrstvy pro ne/zničitelné objekty z objektu Grid.. Proměnné explosionLenght, explosionLifeTime a deathExplosionDistance uchovávají data o explozi, včetně délky exploze, délky života exploze a vzdálenosti, která je pro hráče smrtelná. Tento Skript také vyžaduje některé další pomocné odkazy na skripty, jako jsou DestructiblesCollisions.cs, BombPlant.cs a Gameloop.cs, které jsou získány pomocí metody GetComponent().

i. *Start()* se volá při inicializaci skriptu a získává počáteční délku exploze z metody getSetExplosionLenght() a odkazy na zmíněné pomocné skripty.

ii. *getSetExplosionLenght()* získává počáteční délku exploze z nastavení hry. Pokud taková hodnota nebyla nastavena, použije 1.

iii. *gridCollision()* slouží k detekci kolizí exploze s objekty. Pokud exploze koliduje s objektem, který lze zničit, provádí se zničení objektu, pokrývá se jím poslední exploze a zároveň dojde k obsluze DestroyBox(), která se stará o generovaná vylepšení a ničení boxů.. Pokud exploze narazí na objekt, který nelze zničit, zastaví se. Metoda vrací true, pokud došlo ke kolizi.

iv. *explodeBomb()* vytváří explozi na daných souřadnicích a pokračuje ve směru vektoru exploze dokud nedojde ke kolizi. Tato metoda se volá rekurzivně s nižší délkou exploze, dokud není délka menší než 1.

v. *ExplodeAfterTime()* je podprogram pro vyvolání exploze po určitém čase od položení bomby. Metoda čeká určitý čas a poté provede explozi všemi možnými směry od aktuální pozice bomby. Po provedení exploze se bomba odstraní a zvýší se počet dostupných bomb hráče.

vi. *WasPlayerInExplosion()* zjišťuje, zda se hráč nacházel v explozi. Metoda prochází všechny hráče a exploze a kontroluje vzdálenost mezi hráčem a explozí. Pokud je hráč v určené vzdálenosti od exploze, hráč je deaktivován.

vii. *checkIfEnd()* z objektu Gameloop.cs se volá pro zjištění kolik hráčů přežilo danou explozi.

viii. IncrementExplosionLenght() inkrementuje délku exploze po získání upgrade pro zvýšení délky exploze.

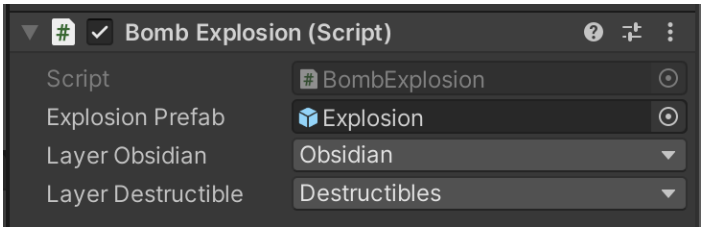


Illustration 23: Implementace BombExplosion.cs

ScoreBoardController.cs se stará o skóre hráčů na herním panelu a nachází se ve stejnojmenném objektu ve scéně GameMap?. Proměnné ObjectP1, ObjectP2, ObjectP3 a ObjectP4 jsou textová pole z rozhraní UI, která zobrazují skóre hráčů na herním panelu. Proměnná Players je pole herních objektů reprezentujících hráče. Proměnná instance je statická reference a slouží k přístupu k tomuto skriptu z jiných skriptů. Proměnná wasIncremented

je logická proměnná, která slouží k zabránění opakovaného inkrementování skóre. Proměnná scores je pole celých čísel, které uchovává skóre jednotlivých hráčů. Proměnná timeBetweenGames určuje časový interval mezi hrami.

i. *Awake()* je volána při inicializaci skriptu a nastavuje instanci.

ii. *Update()* kontroluje, zda-li nedošlo k úmrtí dvou posledních hráčů ve stejným snímku, pak by se spustila newGame().

iii. *newGame()* obnovuje scénu, tedy načte znovu aktuální scénu.

iv. *Start()* inicializuje skóre hráčů na herním panelu. Načítá předchozí skóre uložené v PlayerPrefs. Pro každé textové pole se zobrazí skóre odpovídajícímu hráči. Pokud hráč není aktivní, zobrazí se místo skóre symbol "X".

v. i*nitScores()* načítá předchozí skóre ze PlayerPrefs a ukládá je do pole scores.

vi. *IncrementWinnerScore()* inkrementuje skóre vítěze hry. Inkrementace probíhá pouze jednou v každém kole, aby se zabránilo opakovanému zvýšení skóre.

vii. *SaveLastScore()* ukládá skóre do PlayerPrefs po konci kola.

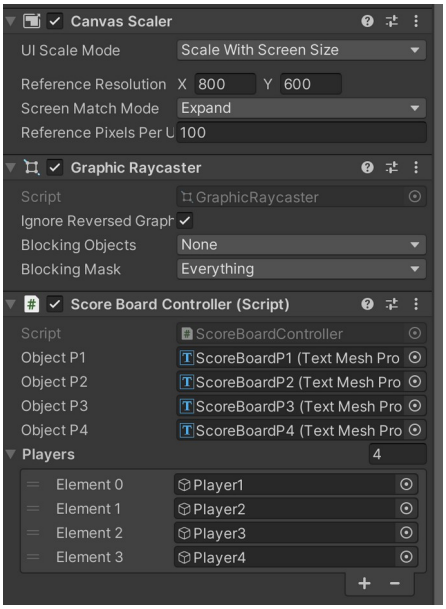


Illustration 24: Implemetnace ScoreBoardController.cs

DirectionAnimation.cs řídí animaci směru pohybu hráče, který je umístěn v jednotlivých podobjektech hráče reprezentující daný směr. Proměnná sr je komponenta SpriteRenderer, která zobrazuje grafiku herního objektu. Pole RunningSprites obsahuje jednotlivé obrázky animace pohybu hráče. Proměnná IdleSprite obsahuje obrázek pro klidový stav hráče. Proměnná currentSprite uchovává index aktuálního obrázku v poli animací. Proměnná nextActionTime

určuje časový okamžik, kdy se má provést další krok animace. Proměnná animationPeriod určuje časový interval mezi jednotlivými kroky animace. Proměnná isIdle je logická proměnná, která určuje, zda je hráč v klidovém stavu.

i. *Awake()* je volána při inicializaci skriptu a nastavuje referenci na komponentu SpriteRenderer.

ii. *Update()* je volána každý snímek a kontroluje stav hráče a aktualizuje animaci. Pokud hráč drží tlačítko pohybu, aktualizuje se animace a přepíná se mezi jednotlivými obrázky animace. Pokud je hráč v klidovém stavu, zobrazí se obrázek pro klidový stav.

iii. *OnEnable()* je volána, když je skript povolen, a zapíná komponentu SpriteRenderer.

iv. *OnDisable()* je volána, když je skript vypnut, a vypíná komponentu SpriteRenderer.

v. *IsIdle()* slouží k změně stavu animace, volá se ze skriptu PlayerMovements.cs.

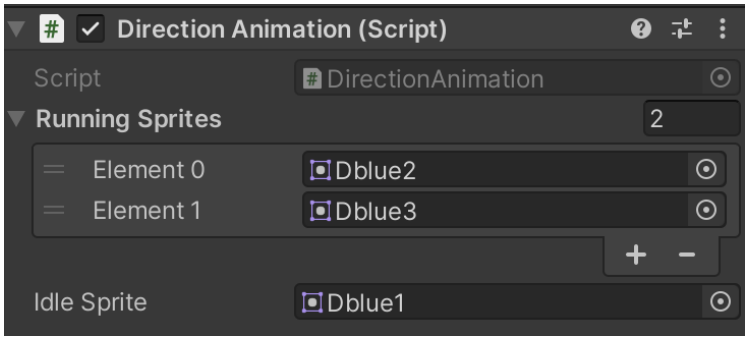


Illustration 25: Implementace DirectionAnimations.cs

DestructiblesCollisions.cs řídí kolize s ničitelnými objekty v hře a je součástí objektu hráče. Proměnná LayerDestructible určuje vrstvu, na které se nacházejí ničitelné objekty. Proměnná Destructible je komponenta Tilemap, která reprezentuje ničitelné objekty ve hře. Pole Upgrades obsahuje seznam herních objektů – Prefabs, které představují různé vylepšení. Proměnná existingUpgrades je seznam herních objektů, které představují aktuálně aktivní powerupy na herní mapě. Proměnná pickupDistance určuje vzdálenost, ve které hráč může sebrat powerup. Proměnná generatingProbability určuje pravděpodobnost generování powerupu. Proměnná players obsahuje seznam herních objektů, které představují hráče. Pomocné proměnné s názvy bombExplosionScript, bombPlantScript a playerMovementScript jsou reference na ostatní skripty, které obsahují veřejné metody pro aktivaci vylepšení.

i. *Start()* je volána při inicializaci skriptu a nastavuje potřebné reference skriptů a hráčů a inicializuje seznam pro aktivní vylepšení.

ii. *GetSetGeneratingProbability()* nastaví hodnotu pravděpodobnosti na implicitní hodnotu, která je modifikována, pokud je v nastavení řečeno jinak.

iii. *Update()* je volána každý snímek a provádí metodu starající se o kolizi vylepšení s hráčem steppedOnUpgrade().

iv. *steppedOnUpgrade()* kontroluje, zda některý hráč stojí na vygenerovaném powerupu a případně zavolá activateUpgrade().

v. *activateUpgrade()* vykonává příslušnou akci daného powerupu s pomocí skriptů s uloženými hodnotami rychlost, počet bomb…

vi. *generateUpgrade()* generuje náhodný powerup s danou pravděpodobností na dané pozici exploze.

vii. *DestroyBox()* provádí logiku ničení ničitelných objektů. Zjišťuje, zda se na pozici exploze nachází ničitelný objekt, a pokud ano, odstraňuje ho a spustí generateUpgrade().

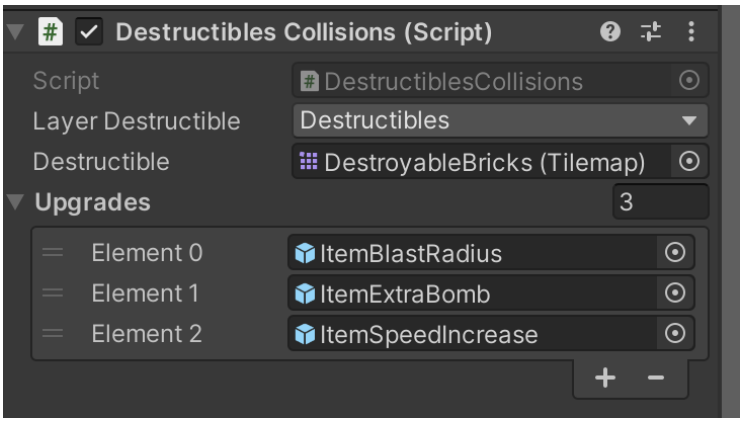


Illustration 26: Implementace DestructiblesCollisions.cs

Gameloop.cs je odpovědný za řízení herní smyčky a událostí v průběhu hry. Proměnná timeBetweenGames určuje časové okno mezi dvěma koly hry. Tento skript slouží k řízení herní smyčky a detekci konce hry, aby mohlo být spuštěno další kolo.

i. *CheckIfEnd()* je volána pro kontrolu, zda došlo ke konci hry. Metoda zjišťuje počet aktivních hráčů ve hře a při jednom zbývajícím hráči zvyšuje skóre v ScoreBoardControlleru pro vítěze a ukládá poslední skóre. Poté je pomocí Invoke a metody newGame() spuštěno nové kolo hry po uplynutí času timeBetweenGames.

ii. *newGame()* resetuje aktuální herní scénu a načítá ji znovu pomocí SceneManageru

VI. Co nebylo doděláno

Potencionálním rozšířením hry je možnost vyvtvořit novou scénu SinglePlayer, kde by si uživatel, v případě absence potřebných hráčů pro lokální multiplayer, mohl zahrát buďto jako sám proti určenému množství hráčů automaticky ovládaných programem, nebo verzi proti příšerám, kde by se sbíraly různé klíče, které by otevíraly skryté dveře do dalších levelů. Zde by se ovšem bylo vhodné přidat „životy“ objektu hráč a různé varianty obtížnosti. Jiným rozšířením je potencionální možnost hraní Bombermana vzdáleně přes server ať už s kamarády či náhodnými soupeři.

VII. Programátorský závěr

Jak bližší seznámení s enginem Unity tak i jednotlivé části vývoje hry pro mne byli zajímavým a zábavným procesem, při kterém jsem se naučil a seznámil s mnoha novými tématy. Ať už návrh grafiky postav, ikony, mapy či rozmýšlení nad implementací metod pro jedlitvé herní objekty. Prostředí Unity je velice intuitivní a skvělý pomocník právě pro návrh i vývoj her. Zajisté mám do budoucna v plánu vytvořenou hru rozšířit o zmíněné možné vylepšení z odstavce výše.