HW01 - C# Defense

Obsah

Zadání:

Za sedmero horami a sedmero řekami žil jednou jeden líný programátor jménem Speed, který rád psal malé herní projekty. Často se však stávalo, že napsal jen krátkou kostru projektu a ve chvíli, kdy bylo třeba uplatnit základní principy OOP, vzdal to a přešel k jinému projektu. Proto je na vás, abyste dokončili zbývající funkcionalitu tower defense hry s názvem C# Defense. Tato funkcionalita zahrnuje logiku obranných věží, projektilů a útočících nepřátel.

Obranné věže

Vaším úkolem bude implementovat 3 typy obranných věží (Basic, Burst, RandomFun), přičemž každá věž má jiné, ale i společné vlastnosti:

Společné vlastnosti:

- Cena
- Dostřel
- Název
- Projektil
- Čas mezi výstřely
- Počet životů

Basic věž:
- Cena (75), Dostřel (20), Název (Basic Tower), Projektil (Basic), Čas mezi výstřely (1.5s), Životy (50)
- Věž si vybere nejbližšího nepřítele, který je na dostřel, a střílí na něj dokud nepřítel neodejde z dostřelu.
- Pokud má věž nějaký cíl, který je na dostřel, věž vystřelí jeden projektil směrem na nepřítele.

Burst věž:
- Cena (150), Dostřel (10), Název (Burst Tower), Projektil (Basic), Čas mezi výstřely (3.0s), Životy (120)
- Věž vybere nepřítele, který je na dostřel a má největší počet životů, a střílí na něj dokud nepřítel neodejde z dostřelu.
- Pokud má věž nějaký cíl, který je na dostřel, věž vystřelí dva projektily směrem na nepřítele, každý projektil se zpožděním 0.2 sekundy.

Random věž:
- Cena (300), Dostrel (10), Název (Random Tower), Projektil (Explosive), Čas mezi výstřely (2.0s), Životy (140)
- Věž vybere náhodného nepřítele, který je na dostřel, a střílí na něj dokud nepřítel neodejde z dostřelu.
- Pokud má věž nějaký cíl, který je na dostřel, věž se rozhoduje. Věž má 20% šanci vystřelit dva projektily (bez zpoždění), 60% šanci vystřelit jeden projektil a 20% šanci nestřílet vůbec.

Komentář:
- Pokud při výběru nepřítele existují dva vhodné cíle (např. oba mají maximální počet životů), nezáleží na tom, kterého si věž vybere.

Obranné projektily

Vaším úkolem bude implementovat 2 druhy projektilů (Basic, Explosive), které budou obranné věže střílet na nepřátele.

Společné vlastnosti:
- Poškození
- Rychlost
- Délka života
- Po zásahu nepřítele nebo po překročení délky života se projektil zničí.

Basic projektil:
- Poškození (25), Rychlost (5), Délka života (5.0s)
- Když projektil zasáhne nepřítele, sníží mu počet životů o 25.

Explosive projektil:
- Poškození (10), Rychlost (5), Délka života (4.0s)
- Když projektil zasáhne nepřítele, sníží mu a nepřátelům ve vzdálenosti 5 počet životů o 10.

Nepřátelé

Vaším úkolem bude implementovat 2 druhy nepřátel (Lazy, Aggresive), kteří se budou snažit dostat na vaši základnu a zničit ji.

Společné vlastnosti:

- Počet životů
- Poškození
- Odměna
- Rychlost
- Po kolizi s věží nebo hradem se nepřítel zničí.

Lazy nepřítel:

- Počet životů (175), Poškození (25), Odměna (25), Rychlost (3)
- Nepřítel postupuje po cestě směrem k poslednímu bodu cesty. Nepřítel postupuje 5 sekund a poté se na 1 sekundu zastaví (za 12 sekund od vytvoření 2 sekundy stál a 10 sekund se pohyboval) a potom opět 5 sekund pokračuje. Tento způsob pohybu se opakuje, až dokud nepřítel nedojde do cíle.
- Když nepřítel narazí na hráčův hrad, odebere hradu 25 životů. Pokud na obrannou věž, odebere jí dvojnásobný počet životů.

Aggresive nepřítel:

- Počet životů (100), Poškození (40), Odměna (10), Rychlost (5)
- Nepřítel postupuje po cestě směrem k poslednímu bodu cesty. Pokud po cestě narazí na obrannou věž, která je ve vzdálenosti menší než 10, začne sa hýbat směrem k věži (vznikne-li mezi časem v okolí nová věž, nepřítel ji ignoruje a pokračuje směrem k původní věži). Pokud je daná věž zničená během toho, co se k ní nepřítel blíží, nepřítel pokračuje dál po cestě s klasickým chováním (hýbe se po cestě a útočí na věže).
- Když nepřítel narazí na hráčův hrad nebo obrannou věž, odebere jim 40 životů.

Komentář:

- Všechny scripty se nachází v adresáři Assets/Scripts. Při vytváření nových ukládejte dané scripty do příslušných adresářů (nové implementace věží do adresáře „Tower“ apod.)

Odevzdávání

Odevzdávat budete stejné adresáře, jaké jste dostali v kostře (UserSettings, ProjectSettings, Packages a Assets), respektive ty adresáře, které nejsou ignorované přiloženým .gitignore souborem. Jiné adresáře neodevzdávejte. Odevzdání nadbytečných souborů může mít za následek malou bodovou ztrátu.

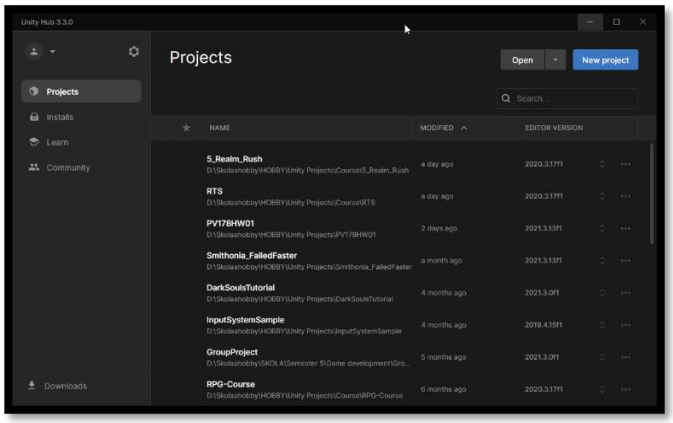
Tyto adresáře odevzdáte tak, jak to bude od vás vyžadovat váš cvičící (buď ve formě zipu do odevzdávárny nebo přes git).

Konec zadání (následuje sekce zaměřená na práci s Unity)

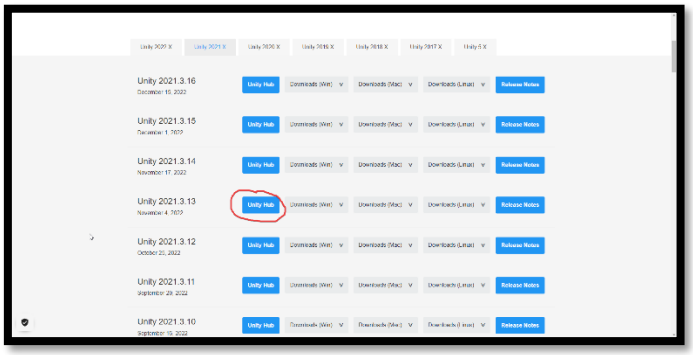
Unity

Pro vypracování této úlohy si budete potřebovat nainstalovat Unity:

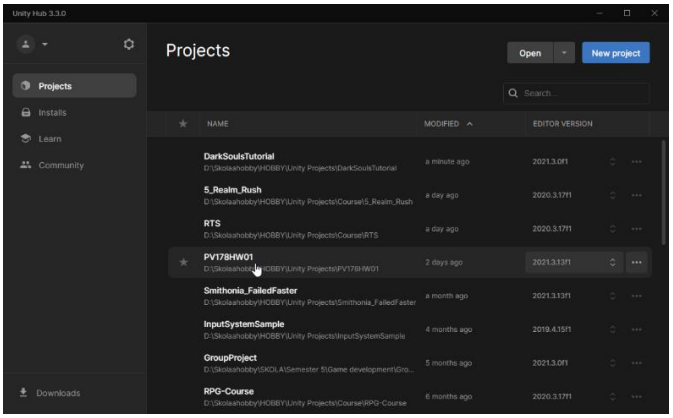
1. Na stránce https://unity.com/download#how-get-started si stáhněte správnou verzi Unity Hub pro váš operační systém a Unity Hub nainstalujte. Takto nějak by měl vypadat váš Unity Hub (akorát bez projektů).



2. Následně si ze stránky https://unity.com/releases/editor/archive stáhněte Unity Editor (verziu 2021.3.13) kliknutím na tlačítko „Unity Hub“.



3. Po kliknutí se vám zobrazí instalační okno, ve kterém vyberete „Microsoft Visual Studio Community ….“ Dev Tools.
4. Po nainstalovaní Unity Editoru v Unity Hube si otevřetete kostru domácího úkolu kliknutím na tlačítko „Open“ a vybráním adresáře s Unity projektom.
5. Poté, co se Unity projekt načte do Unity Hubu, ho můžete otevřít kliknutím.



V případě, že se vám nepodaří zprovoznit Unity na vašem zařízení, můžete využít školní počítače, na kterých je už nainstalovaná správná verze Unity. K těmto počítačům se můžete připojit přes Remote Desktop (https://www.fi.muni.cz/tech/win/classrooms.html#rdesktop) a domácí úkol tak vypracovat. Úkol si musíte otevřít na lokálním disku C:/Users/x(jméno) a po ukončení práce si projekt uložit do vašeho remote git repozitáře.

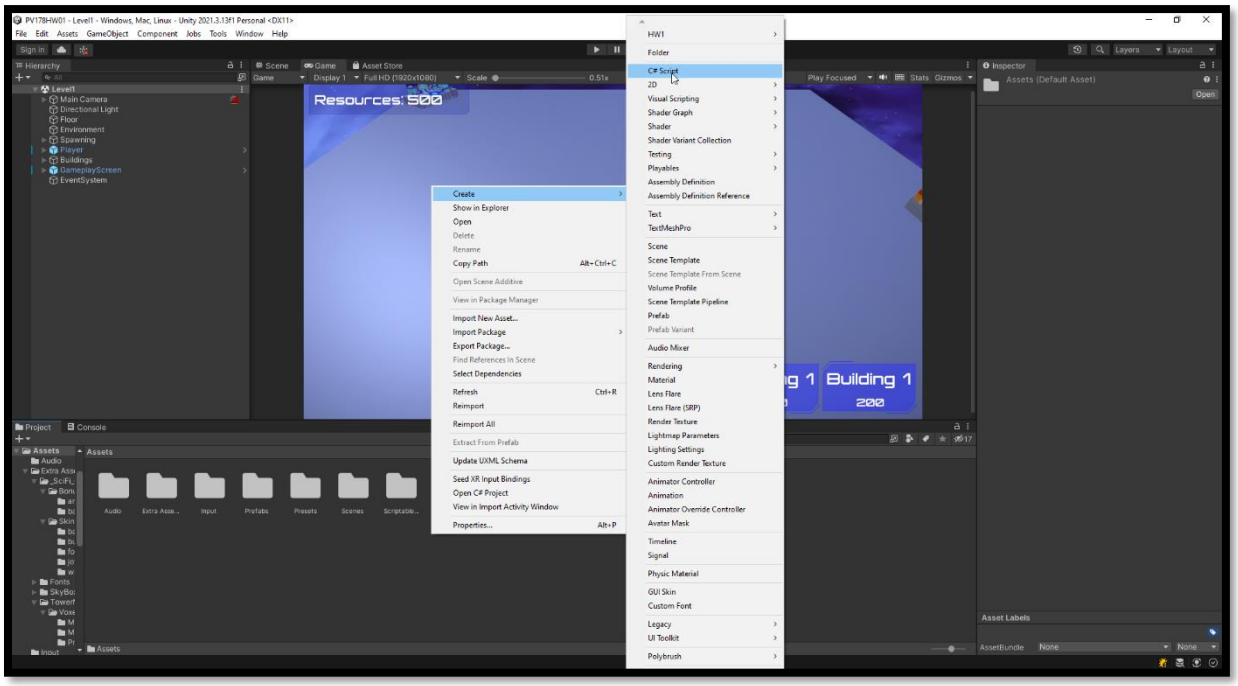
Unity Helper

Na implementaci tohoto domácího úkolu budete muset využít různou funkcionalitu, kterou vám Unity a Unity Editor nabízí.

Unity Editor:

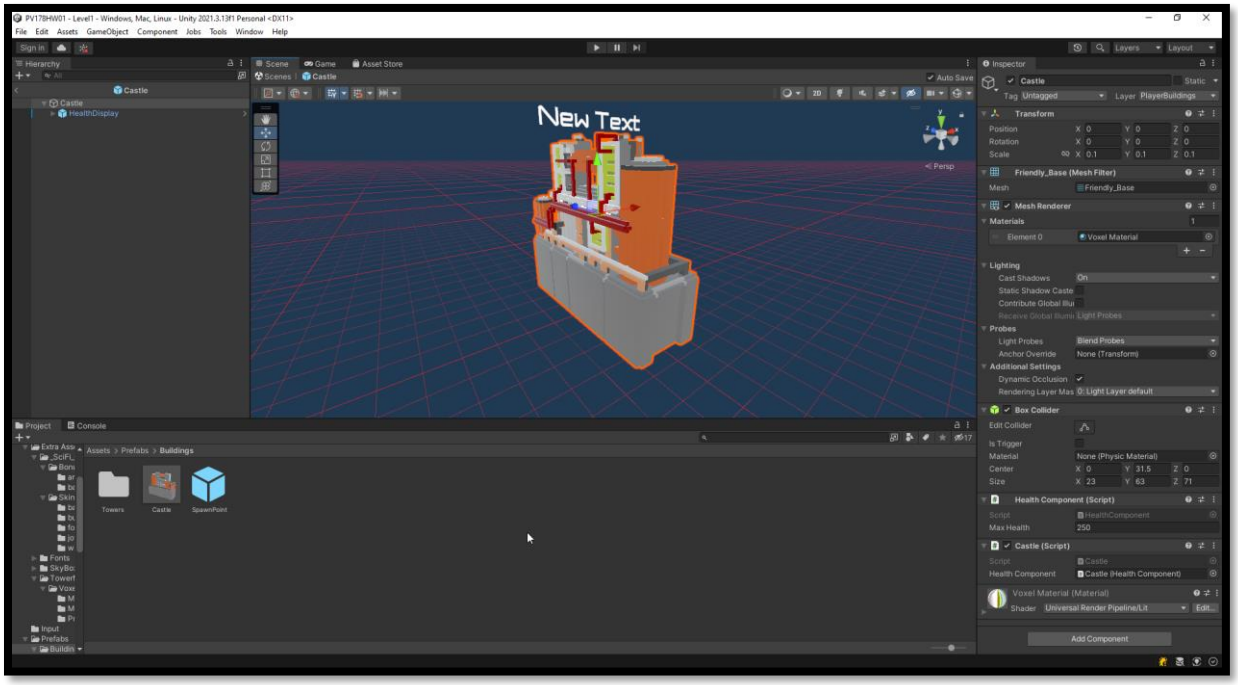
Vytváření C# class:

- Pro vytvoření nové třídy v Unity klikněte pravým tlačítkem mezi adresáře a vyberte možnost C# Script.
- Všechny classy, které tímto postupem vytvoříte, budou dědit ze speciální classy MonoBehaviour. Tato classa nám umožňuje vytvořenou classu přidat do Prefabu a zároveň nám umožňuje přístup k různým Unity callbackům.



Prefabs (Pre-fabricated objects):

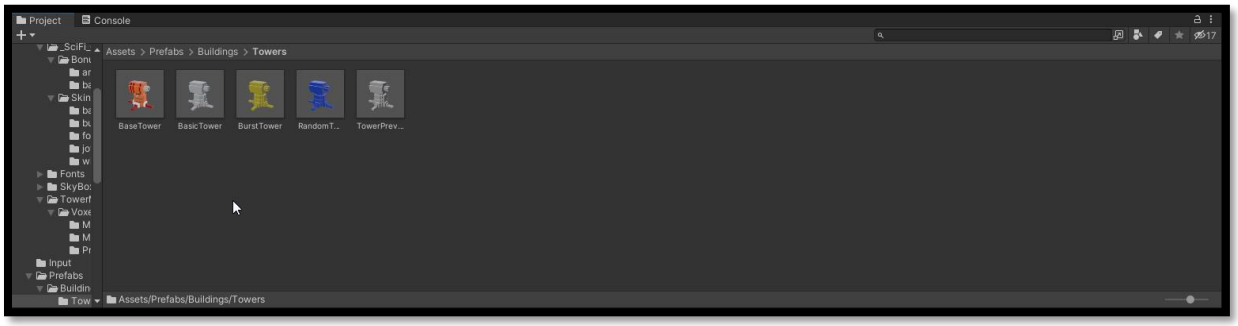
- Prefaby v Unity jsou templaty uložené v souborovém systému, z kterých můžeme během run-timu vytvářet kopie, které vkládáme do herního světa.
- V této úloze nebudete muset vytvářet nové prefaby, ale budete muset daným prefabom přidávat classy, které během implementace vytvoříte.



- Pro přidání classy k prefabu klikněte na prefab, který chcete modifikovat. Na pravé straně se objeví detail prefabu. V detailu nascrollujete dolů k tlačítku „Add Component“, kde vyberete název classy kterou chcete pridat.
- Pro odstranění classy z prefabu klikneme pravým tlačítkem na název classy v prefabe a vybereme možnost „Remove Component“.

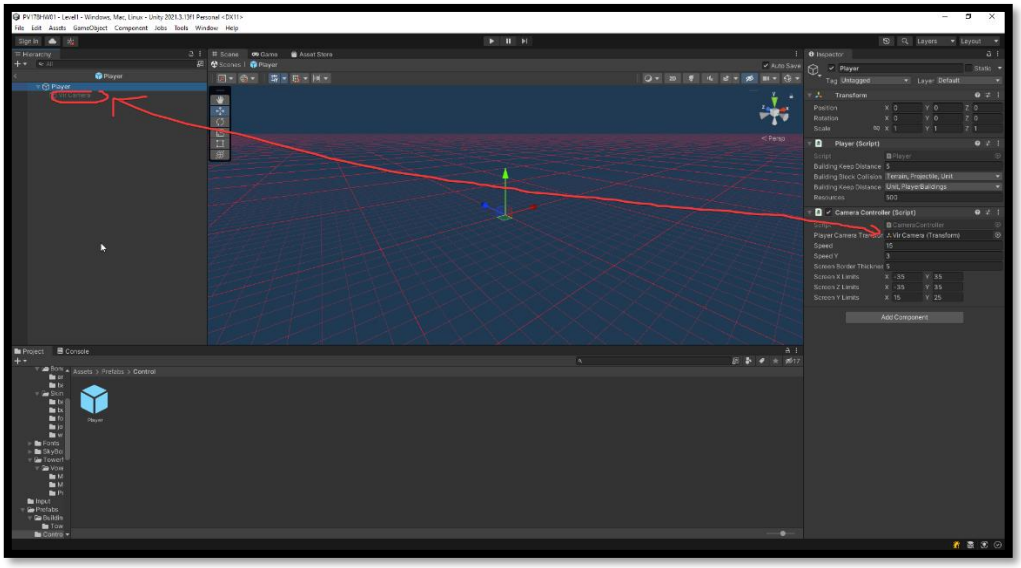
Předpřipravené prefaby:

- V projektu jsou pro vás už předpřipravené prefaby pro jednotlivé druhy věží, projektilů a nepřátel v souboru „Prefabs“. Tyto prefaby modifikujte (nevytvářejte nové a nemažte je) přidáváním class, které vytvoříte (nemodifikujte ale prefaby s názvem v tvaru „Base.....“).

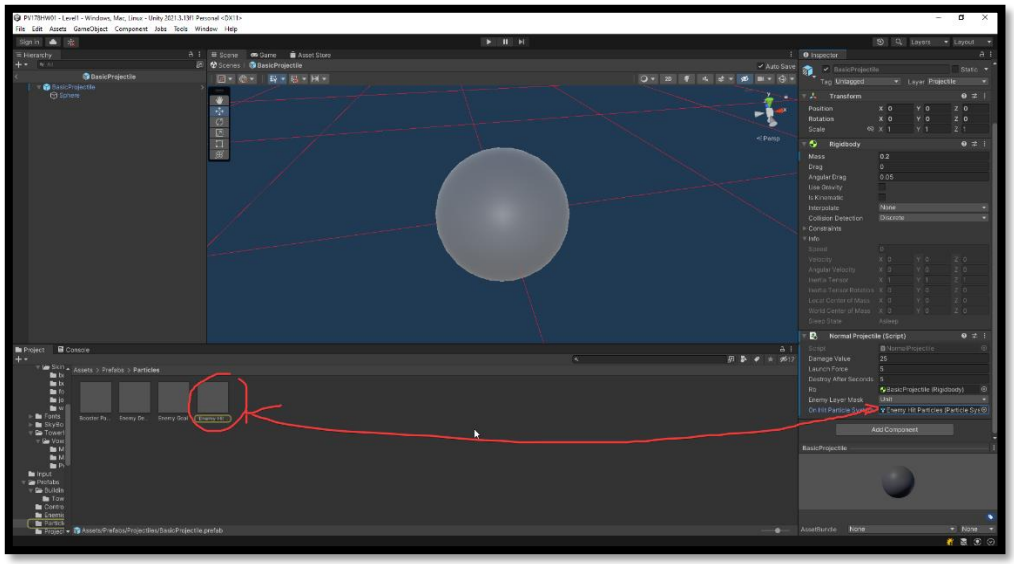


Linkování objektů:

- V kódu některých class můžete vidět atributy „[SerializeField]“. Tento atribut nám umožňuje nastavování prvotních hodnot jednotlivých fieldů v classe pomocí Unity Editoru.
- Fieldy, které referencují nějakou classu můžeme referencovat tak, že potáhneme daný objekt z hierarchie vlevo do fieldu dané componenty vpravo. Takto referencujeme objekt, který se nachází na stejném prefabu.

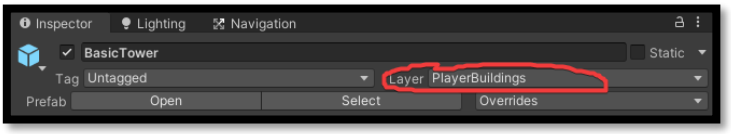


- Fieldy můžou referencovat i jiné prefaby (věž bude referencovat prefab projektilu, který bude střílet). Postup referencování je stejný (potáhnete prefab do fieldu, který chcete referencovat).



Physics layers:

- Na to aby se správně vyhodnocovaly fyzikální operace jako kolize, je nutné jednotlivým objektům přiřadit fyzikální vrstvu. Naštěstí všem objektům už byly tyto vrstvy přirazeny.
- Některé classy, jako Tower, Projectile a Enemy, ale vyžadují specifikaci jednotlivých layers přes LayerMasks (Tower potřebuje vědět na jaké layers má útočit a podobně).
- Zkontrolujte si v prefabech, na jakých layerech se jednotlivé GameObjecty nacházejí a podle nich specifikujte jednotlivé LayerMasks.



Metody:

Při programování budete potřebovat několik metod a komponent, které jsou pro vás předvytvořené, nebo jsou specifické pro Unity:

Custom komponenty:

- MovementComponent
  - Custom classa pro navigování nepřátel.
  - *MoveAlongPath()*
    - Přikáže nepříteli, aby pokračoval dál po cestě. Tuto metodu není nutné volat každý Update.
  - *MoveTowards(Tranform target)*
    - Přikáže nepříteli, aby se pohyboval směrem k cíli. Vráti true, pokud je cíl dosažitelný. Tuto metodu není nutné volat každý Update.
  - *CancelMovement()*
    - Přikáže nenepříteli, aby se přestal hýbat.
- HealthComponent
  - Udržuje uvnitř počet životů daného objektu.
- Transform
  - Speciální class která nám umožňuje práci s pozicí a rotací objektu.
- Rigidbody
  - Speciální class která simuluje fyzikální jevy na objektu jako rychlost, akceleraci a gravitaci.
- Vector3
  - Speciální struct který nám umožňuje práci s vektory jako výpočet vzdálenosti a podobně.
- Time
  - Unity statická class která nám umožňuje práci s časem

Unity metody:

- *Update()*
  - Callback, který je volaný Unity každý snímek (snímek si představte pod pojmem vykreslení obrazu (v hrách s 60 FPS by se zavolala metoda Update 60krát za sekundu))
  - Obsahuje často hlavní logiku classy (chování věží nebo nepřátel).
- *Start()*
  - Callback, který je volaný Unity na začátku života daného objektu. Když je objekt vytvořený, v Start callbacku dochází často k inicializaci, výpočtu anebo cachování prvotních dat.
- *OnCollisionEnter(Collision other)*
  - Callback, který je volaný Unity na objektu v případě, že došlo ke kolizi s nějakým jiným objektem (kolize mezi nepřítelem a věží/hradem).
- *OnTriggerEnter(Collider other)*
  - Callback, který je volaný Unity na objektu v případě, že došlo ke kolizi s nějakým jiným objektem (kolize mezi projektilem a nepřítelem). Colliders jsou ale nastavené jako triggers (nepodléhají fyzickým kolizím).

- *GetComponent<>()/TryGetComponent<>()*
  - V Unity pracujeme s GameObjects, které si můžete představit jako seznam různých objektů, které dědí z MonoBehavior (všechny classy, co jste mohli vidět výše jako součást prefabů) (prefaby jsou pouze uložené GameObjects v souboru).
  - Tuto metodu můžeme použít pro získání nějakého objektu na daném GameObjectu jako: other.gameObject.GetComponent<Castle>().
- *Instantiate(…)*
  - Unity nepodporuje vytváření objektů, které dědí z MonoBehavior pomocí klasického C# konstruktoru. Místo něj se používá metoda Instantiate zděděná z MonoBehavior classy, která vytvoří GameObject společně s všemi objekty, které má zaregistrované v prefabu.
- *Destroy()*
  - Unity destruktor z MonoBehavior, který umožňuje zničit nejen jednotlivé (MonoBehavior) objekty, ale i celý GameObject (při zničení GameObjektu se automaticky zničí i všechny objekty, které obsahuje (MovementComponent, HealthComponent…).
  - V případě, že měl nějaký objekt referenci na objekt, kterí byl zničený, tak se daná reference změní na null.
- *Physics.SphereCastAll()/Physics.OverlapSphere()*
  - Zjistí, které všechny objekty se nachází v určité oblasti.

Komentář:

- V této sekci jsou popsané metody, které určitě budete potřebovat, poměrně zjednodušeně. V případě, že vám tyto vysvětlení nestačí, anebo jsem na něco zapomněl, doporučuji Unity dokumentaci https://docs.unity.com/.