Prefabs 🡪 každý gameObject se může uložit jakožto Prefab, předvytvořený objekt, který se bude ve scéně opakovat 🡪 PlayerBullet

Time.DeltaTime 🡪 počet snímků za sekundu = na různých počítačích se bude pohybovat vše stejně rychle

Edit Layers 🡪 co má interagovat s čím – NEŘEŠÍ SE KOLIZE pouze, jestli objekt má nějak interagovat = nemusíme tedy řešit v kódu

GameObject Manager 🡪 managment hry, například životů = při zničení hráče nemá smysl ničit i Health systém, proto se tento systém vytváří v odděleném GameObject.  
Nepřátelé mají svůj vlastní health systém, protože jsou to individuální gameobjects, takže není nutné, aby se pro ně vytvářel zvlášť jako u hráče.

Pořadí v Hierarchii tvoří Scene graph který říká rendereru v jakém pořadí se má vykreslovat scéna 🡪 spodní objekty se překreslí přes již vykreslené

Vector3.Lerp() 🡪 linearní interpolace viz grafika

Tilemap collider v kombinaci s Composite Collider 🡺 celá mapa je považována za jeden Collider

Collisionboxy s isTrigger = true 🡪 musí se nastavit Layers s čím mají interagovat (například, aby ignorovali střely)

StartCoroutines() 🡪 <https://docs.unity3d.com/Manual/Coroutines.html>

Image RaycastTarget uncheck 🡪 lze „prokliknout“ k UI, myšleno, při překreslení tlačítka průhledným obrázkem lze stále klikat na tlačítka

K designu: esc klávesa 🡪 pause, nikdy se nevyužívá jako třeba vypnutí hry (hráč očekává pause)

Vnitřní třídy nejsou vidět 🡪 musí se přidat anotace [System.Serializable] (dědičnost z monobehavior tvoří komponentu, takže nutný samostatný script)

Anotace [Header] blok kódu pod anotací (dokud není mezera) má v unity inspectoru nastavenou hlavičku a oddělena do sekcí

Culling mask v kameře 🡪 určení co kamera vlastně může vidět

RenderTexture v kameře 🡪 soubor, do kterého se ukládá, co zrovna kamera vidí 🡪 využití například v UI komponentě raw image (oproti klasickému Image nepoužívá sprite, ale texturu) = použítí k minimapě