# Tower Defense – Technická dokumentácia

Hra Tower Defense pozostáva z troch časti,  alebo lepšie povedané z troch projektov, ktoré spolupracujú na to aby hra fungovala. Keďže ide o hru pre dvoch hráčoch, ktorý hrajú hru po sieti, takže to využíva architektúru server – klient, kde je jeden server , ktorý sa stará o napájanie klientov na neho a taktiež o cely priebeh hry. Na serveri beží cela hra, cela jej logika je na serveri, ten iba posiela obom hlúpym klientom informácie na to aby vedeli ako renderovať hru. Klienti na druhú stranu spracúvajú vstupy od hráčov, ktorý klikajú myškou na mapu a vyberajú si z menu akcie, ktoré chcú vykonať. Server je TCP listener a klienti sú TCP klienti. Dáta sa posielajú cez TCP spojenie. Posledná časť je knižnica, ktorú aj klient aj server používajú. V nej je komunikačný protokol vďaka ktorému si vedia server a klient medzi sebou posielať dáta potrebne na to aby hra fungovala.

## TowerDefenseNetworking knižnica

Hlavná trieda tejto knižnice je CommunicationProtocol. Tá si vyžadujú pri inicializácii predať ako argument TCP klienta. To reprezentuje klienta, ktorý sa napojil na server a teda od neho si vyžiadame network stream. Ide o stream do ktorého vieme zapisovať a na druhej strane siete čítať dáta. Stream funguje jednoducho a to tak ,že zapisujeme do neho pole bytov a následne na druhej strane čítame pole bytov. To čo znamenajú jednotlivé byty už rieši CommunicationProtocol. Ten obsahujú konštanty, ktoré definujú veľkosti jednotlivých hlavičiek dát, ktoré budeme po sieti posielať. Naša trieda obsahuje privátne metódy. ktoré sú pomocné a využívajú sa pri verejných metódach. Sú to metódy ako zápis bytu, intu, doublu, stringu do  nášho streamu a takisto prečítanie jedného bytu, intu, double a stringu. Tieto uvedene metódy používajú ďalšie pomocne metódy , ktoré dôkazu danú hodnotu previesť do poľa bytov. Posledná privátna metóda je samotne poslanie dát cez stream, teda každá verejná metóda ktorou úlohou je poslať nejaké dáta, tak na samotnom konci zavolá tuto metódu a predá jej pole bytov, ktoré vytvorila a naparsovala, ktoré potom daná privátna metóda zapíše do streamu. Teraz sa poďme venovať posielanú herných dát. Network stream je TCP spojenie a teda dáta sa posielajú ako prúd ale toto spojenie nám samo osebe zaisťuje to aby dorazili jednotlivé byty v takom poradí ako ich posielame, tak isto neriešime overovanie dát ci dorazili a ci sú neporušene o to sa všetko stará už network stream.

My posielame bloky dát, na začiatku každého bloku dát posielame jeden byt ktorý reprezentuje packet type , teda to nám určuje, čo reprezentuje cely blok dát alebo posielame iba daný packet type čo reprezentuje keď posielame flagy , teda iba krátku informáciu , ktorá nemá telo/náklad/ payload.

Keď posielame dáta a teda zapisujeme do siete tak na začiatku každej tejto metódy sa zavolá metóda, ktorá vytvorí prázdne pole bytov už správnej veľkosti, pretože ako argument berie veľkosť poľa. Na začiatok pridá už typ paketu a vytvorí offset storer triedu. Keďže do poľa zapisujeme tak treba si nám pamätať ,kde už sme čo zapísali, respektíve zapisujeme za sebou takže stačí si nám pamätať kde momentálne zapisujeme v poli na akom indexe. Na to nám slúži tato trieda, ktorá si pamätá tento index a stále keď voláme metódu na zápis nejakých dát tak predáme ako argument aj tuto triedu ,ktorá inkrementuje svoj index podlá toho čo sa do poľa bytov zapísalo.

Ešte pred tým ako sa budeme venovať jednotlivým metódam, cez ktoré posielame údaje o hre tak ostáva ešte jedna metóda, ktorej úlohou je precitať len prvý byt, rozhodnúť aký je tam packet typ a tuto informácia predať klientovi alebo serveru, aby už vedeli aku metódu majú zavolať pre precitanie zvyku bloku dát.

Teraz uvedieme metódy, ktoré posielajú dáta o hre. Každá tato metóda je v dvojici. Teda jedna je na zapíš, ktorá berie ako argument dáta, ktoré chceme poslať a druha v pare je na čítanie zo streamu ktorá vracia dáta, ktoré sme poslali.

Message je metóda ktorou pošleme textovú správu, funguje to na princípe, že konvertujeme string do poľa bytov a údaj o veľkosti tohto poľa zabalíme do dátového bloku aby sme pri čítaní vedeli aký veľký úsek zo streamu mame precitať aby sme získali celu textovú správu. Správy si posielajú klienti keď si píšu v chate , teda klient napíše niečo do chatu a odošle správu, ta príde na server a server to pošle obom hracom.

Tower Action, keď hráč chce postaviť vežu alebo ju vylepšiť poprípade zničiť tak na strane klienta vykoná danú akciu užívateľ, klient to zabalí do štruktúry tower action, ta v sebe obsahuje informácie ako na akej lokácii chceme postaviť vezú , poprípade že na týchto súradniciach chceme vezú vylepšiť alebo zničiť,  typ akcii teda ci sa bude jednať o stavanie, vylepšovanie alebo ničenie , typ samotnej veži. Keď chce hráč stavať nech vie server  akú vežu treba postaviť , a posledné je ID hráča aby server vedel aký hráč chce danú akciu vykonať.

Monster Action. Hráč chce poslať príšeru na nepriateľa tak tato metóda pošle cez sieť informáciu, že aký hráč a aký druh príšeru chce poslať.

Flag. Ide o poslanie samotného packet typu , ktorý ma niečo reprezentovať a to napr. connection flag, ktorý posiela server klientovi aby vedel ci tam klient stále je a ma so serverom spojenie , flag ktorý  klient pošle keď je hráč pripravený už začať hru a server potom vie že už ma zapnúť hru alebo flag, ktorý pošle server klientovi, že sa druhy klient odpojil, stratil server s nim spojenie a teda da druhému vedieť, že je koniec hry z dôvodu odpojenia druhého hráča.

World Map jednoducho pošle udaj o celej mape a to aj veľkosti mapy aby sme vedeli koľko mame precitať bytov a ako mame pri čítaní napasovať jedno rozmerne pole do dvojrozmerného, v ktorom je reprezentovaná herná mapa.

PlayerState pracuje so statusom hráča , teda štruktúrou v ktorej sú údaje o hráčovi a to jeho život , koľko ma goldov a udaj a nepriateľovom živote. Tuto metódu server posiela klientom vždy keď sa zmenia údaje hráčovi, teda príde o život alebo získa peniaze.

PlayerID. Server pošle na začiatku informáciu klientovi aby vedel klient aké ma daný hráč ID.

Tower a Monster Stats. Metódy, ktoré pošlú údaje o príšere alebo vežičke. Sú to štruktúry v ktorých sú údaje ako život, rýchlosť , sila atď.

Monster State. Pošle list v ktorom sú render info o príšerách, teda informácie ktorú posiela server klientovi, vždy keď server updajtuje hru a posunie, pohne príšery, tak pre všetky príšery vytvorí štruktúru, ktorá ma všetky potrebne údaje o tom aby vedel na základe nej klient vykresliť príšeru na obrazovku. Ide o lokáciu, typ príšery nech vie klient vykresliť správny obrázok a hodnotu koľko ostáva života príšere v % hodnote.

GameEndStatus metóda, ktorá posiela údaj o tom, že hra skončila a ktorý hráč je víťaz. Metoda posiela status teda enum, či je hráč víťaz alebo porazený. Server ak detekuje, že skončila hra tak pošle hracom tento status.

Všetky tieto štruktúry a enumy, ktoré sa používajú v týchto metódach sú definovane v knižnici.

Spomenul by som ešte tower stats. Tie sa dajú vylisovať a poďme sa pozrieť na princíp. Položme si najprv dva problémy. Jeden je či vieme ešte vylepšiť staty a druhy ako ich vylepšime. Tieto dva

problémy riešia rozhrania a následne konkrétne triedy ktoré implementujú tieto rozhrania. Otázku ci vieme vylepšiť rieši TowerUpgradeChecker. Ten funguje na princípe levelu veži. Teda ak je veža už maximálny level tak nedovolí statom aby sa vylepšili. TowerDefenseRatio zas na základe ratio hodnoty vynásobí staty ak sa môžu vylepšiť.

# Klient

Klient sa skladá z nasledujúcich častí:

**ClientFrom**

Obsahuje všetky panely, ktoré reprezentujú UI hry, teda menu kde si vie hráč vybrať akú akciu chce vykonať, teda napr. postaviť vežu alebo poslať príšeru. Tieto panely v sebe obsahujú tlačidla a kazde z nich vykonávaj určitú akciu. Ako príklad uvedieme panel, ktorý sa zobrazí keď hráč chce poslať príšery a teda klikne na dane tatíčko reprezentujúce danú príšeru a to pošle cez klienta na server monster action do ktorej zabalí aký druh príšery hráč posiela.

Takisto tu nájdeme ak rôzne text area, labely a picture boxy ktoré následne hra updajtuje na základe toho aký je stav hry. Napr. tu nájdeme metódu ktorá berie status hráča a ten potom naparsuje do lablov ktoré ukazujú v hre staty o hráčovi ako je jeho život a podobne. Množstvo metód sa stará len o to čo ma byt hráčovi na obrazovke viditeľne, teda napr. ak hráč ma zobrazenú menu pre akciu stavanie veží a myšou vstupy do priestoru tlačidla tak sa v tomto menu objavia informácie o veže. Teda zavolá sa metóda, ktorá nastaví týmto prvkom viditeľnosť aby boli viditeľne a následne do nich uloží údaje o správnej veže. Keď odídeme z lokácie tlačidla tak sa zavolá metóda aby všetko nastavila opäť neviditeľným.

Existuje tu aj timer, ktorý je nastavený tak aby tikal 60X za sekundu a vždy keď tikne tak vymaže to celu grafiku na obrazovke a teda sa automaticky zavolá metóda na vykreslenie, ktorá už zavolá samotnú hru aby sa vykreslila a predá jej grafiku, do ktorej vie hra vykresľovať objekty, od winformu.

Keď sa klient zapne sa tento form a do text area sa vyplnia údaje o IP adrese serveru a portu. Následne vie užívateľ kliknúť na tlačidlo connect, čo vytvorí nového TCP klienta ktorý sa napojí na server. Ak je tu úspešne form nastaví tlačidlo pre začatie hry na viditeľné a hráč jeho stlačením pošle informáciu ako flag, že je pripravený začať hru.

Keď je klienta napojení na server tak vie zadávať textové správy do textového priestoru a stlačením tlačidla Send sa vyberie text, čo zadal a pošle sa na server, ten to následne prepošle obom hracom a daný text sa potom cez správnu metódu appendne do textového priestoru reprezentujúceho chat.

Posledná vec k form je, že ak sa začne hra teda server začne posielať klientovi údaje o hre tak ak pošle hernú mapu tak na základe toho sa zavolá metóda, ktorá podľa veľkostí mapy zväčši herne okno a nastaví všetky panely tak aby sa zobrazovali v strede hernej obrazovky.

**Client**

Ide o triedu , ktorá v seba ma CommucinationProtocol , teda spojenie so serverom. Pri vytváraní si berie referenciu na client form a samotnú hru aby vedela na týchto objektoch volať dane metódy.

Zavolaním metódy Connect, ktorá sa vola z client form keď užívateľ klikne na tlačidlo connect, sa vytvorí TCP client , ktorý sa pokúsi napojiť na server a nastaví to že klient beží, vytvorí to komunikačný kanál so serverom, ktorému predá TCP klienta a spusti sa Task v ktorom bude bežať privátna metóda Run, ktorá najprv inicializuje akcie a potom dokým je klient nastavený, že beží tak v

tomto tasku prebieha while loop , v ktorom klient čaká kým budú nejaké dáta v streame od serveru a následne to za pomoci triedy CommunicationProtocol prečíta packet typ a na základe toho sa zavolá vhodná akcia, čo je funkcia ktorá ma spracovať prichádzajúce dáta zo siete. Napr. Uveďme par príkladov ako ,že príde zo serveru udaj o stats príšery tak si to klient uloží .Príde informácia o mape tak klient zavolá na hre metódu pre načítanie hernej mapy a predá jej prečítanú mapu zo siete .Ako posledné uveďme, že príde novy list obsahujúci informácie o renderovaní príšer a ta akcia v hre prepíše údaje aby hra keď sa bude znovu vykresľovať už vykreslila príšery v nových stavoch.

Tato trieda obsahuje aj metódy na posielanie dát cez sieť takže ak client form zachytí kliknutie myši a to sa v hre vyhodnotí ako tuchnutie pre menu na stavanie veži a hráč si vyberie danu akciu tak sa kontaktuje tato trieda client a zavolá sa metóda na poslanie tower action, ktorú dostane už vytvorenú od hry.

**Game**

Ide o samotnú hru , ktorá sa nevie updajtovat. Jej úlohou je len renderovať hernú mapu , príšery a kruhy dosahu veži, teda aby hráč vedel kde ma veža dosah.

Ako už bolo povedane najdôležitejšia metóda hry je Draw, ktorá je zavolaná z client formu. Tato metóda na hernej mape zavolá vykreslenie, čo vykresli hernú mapu a následne pre každú príšeru v zozname na renderovanie zavolá metódu na samotne vykreslenie príšery, ktora vykresli priseru na základe render info a vytvorí nad príšerou aj HealtBar, reprezentujúci život príšery.

Hra ma aj metódy , ktoré zavolá client, keď dostane od serveru niejaku akciu na spracovanie danej akcie, teda napr. Ak dostaneme akciu o tom, že sa vytvorila nová veža tak sa zavolá na hernej mape metóda na postavenie veže.

Hra na strane klienta funguje tak že nie každý frame je posielaný cez server. Server posiela iba ak sa niečo na mape zmení a pošle iba tu jednu zmenu a tak sa hra na klientovi zariadi. Server posiela iba stále nove údaje o príšerách to kde sa nachádzajú. Ak príšera zomrie tak server už nepošle jej renderovanie údaje a teda ju hra na strane klienta už nevykresli

Keď užívateľ klikne myšou na nejaký game tile, čo je herne políčko tak z winform sa zavolá na hre metóda , ktorá zapracuje toto kliknutie a to tak že od objektu hernej mapy  si popýta tile na ktorý sa kliklo, ale najprv overí ci existuje tile na tomto mieste, pretože môžeme kliknúť aj inde ako je mapa, a zavolá sa akcia na danom tile.

**GameMap**

Reprezentuje celu hernú mapu, teda ma v sebe dvojrozmerne pole v ktorom sa nachádzajú  tile objekty, čo reprezentujú jednotlivé herné políčka.

Pri vytváraní tohto objektu sa predá argument ktorý reprezentuje mapu v podobe ako sa posiela cez sieť a server ju poslal klientovi. Na základe toho vytvorí hernú mapu.

Poskytuje metódy na overenie ci políčko na týchto súradniciach exituje, metódu ktorá vráti konkrétne políčko.

Obsahuje slovník v ktorom sú uložené veže, teda ako kľuč je lokalita políčka a hodnota je samotný objekt veže. Vďaka tomu ma mapa metódu, ktorá vráti konkrétnu vežu na základe predania ako argumentu lokalitu.

Mapa ma všetky metódy, ktoré riešia zmenu mapy a teda metódu na vytvorenie novej veže, na zničenie veže, teda na danom mieste sa zaznačí že je tam opäť prázdne políčko a zo slovníka sa odstráni údaj o tom že exituje na takom mieste veža.

Taktiež ma metódu Draw, ktorá zavolá pre každé políčko v dvojrozmernom poli metódu draw toho políčka a následne to zavolá pre každú vežu v zozname metódu na vykreslenie kruhu okolo veži čo reprezentuje dosah útoku veže.

**Tile**

Reprezentuje samotne políčko. Poskytuje properties ako aký hráč vlastný políčko. To sa využíva pritom keď hráč klika tak ak klikne nie na svoje políčko tak nech sa nestane nič aby zbytočne sa neposielalo na server či môže hráč kliknúť na dane políčko. To pridáva klientovi to že neje až taký hlúpy a par veci vie aby uľahčil chod fungovania hry. Ďalej vie kde sa nachádza, ma v sebe Bitmap objekt , čo je vlastne obrázok políčka a ma v sebe udaj že ak sa na neho klikne tak na základe enum typu vie neskôr client form otvoriť konkrétne UI menu. Teda políčko kde vieme postaviť vežu ma v sebe enum typ podľa ktorého sa zapne menu na stavanie veži a políčko na kupovanie a posielanie príšer ma typ aby sa otvorili menu pre príšery.

Metódy sú tu Draw , čo vykresli políčko a action čo zisti ci dane políčko vlastný hráč a ak áno tak to zavolá na hre metódu ktorá otvorí menu na základe políčka.

**Tower**

Veža ma v sebe properties ako staty veže , objekt TowerRangeCircle, (čo reprezentuje kruh okolo veže. Ide o pomocnú triedu ktorá na základe vstupných argumentov pri vytváraní a to je stred kružnice a farba , ktoré dostane od hernej mapy, vykresľuje kružnicu podlá toho aký ma veža dosah. Tento objekt ma dve metódy a to jednu na samotne vykreslenie kruhu a metódu, ktorá dostane novy dosah a vykresli kruh väčším , to sa zavolá keď sa veža vylepši.), typ veže , teda čo je to za veža, a objekty ktoré sa volajú keď sa zavolá metóda na vylepšenie statov, a to sú objekty na zisťovanie či sa danej veže dajú staty vylepšiť a samotný objekt, ktorý vylepši staty. Tieto objekty už boli popísane v časti o knižnici.

# Server

Server tvoria nasledujúce triedy:

**ServerForm**

Ide o winform, kde nájdeme tri text area, jednu pre IP adresu a druhu pre číslo portu, na ktorých bude bežať server. Tlačidlo ,ktoré zapne server, teda zavolá metódu na objekte Server, a tlačidlo ktoré na tomto objekte zavolá metódu aby sa server zastavil. Nájdeme tu ešte jednu metódu, ktorá berie textovú správu ako argument a dopise text do tretej text area, ktorá ma reprezentovať info , ktoré vypisuje objekt server o tom ako pracuje. Napr. Sa tam vypisuje info, keď je hráč pripravený.

Taktiež je tam list box kde sa vyberá herná mapa, ktorú potom server pošle klientom. Ak neje zvolená tak sa server nevytvorí a ServerForm požiada užívateľa aby zvolil mapu.

**Server**

Server ma za úlohu riešiť dva veci. Jedna je spracovanie a prijímanie prichádzajúcich žiadosti o spojenie cez sieť od klientov a druhu ma za úlohu hostovať celu hru, teda ju updajtovat, riešiť logiku hry.

Sever je stavový vo viacerých formách, jeden stav reprezentuje ci server beží. Na to je enum ServerStatus, ktorý nadobúda hodnoty OFF a ON. Pre tu nájdeme metódu, ktorá vracia true podlá toho ci server beží. Ďalší stav udáva v akom je server stave, čo sa týka napájania hráčov, teda či server čaká na prvého hráča, druhého hráča alebo už je server plný. Tento stav je reprezentovaný ako State pattern, teda mame rozhranie IServerState ktoré ma metódu do action a berie argument referenciu na objekt server aby sme mohli k nemu pristupovať a potom tri samotne implementácie, ktoré reprezentujú dane spomínane tri stavy . V každom tomto stave robíme iný typ akcie a na konci zmeníme na novy stav.

Metoda StartServer, ktorá je zavolaná z ServerForm stlačením tlačidla štart, dostane IP adresu a port a následne vytvorí nového TCPListener, čo reprezentuje server v sieťovom spojení, nastaví stav serveru na to, že server čaká na prvého hráča a vytvorí dva nove tasky. V jednom bude bežať metóda StartListenForConnectionRequests, ide o while loop dokým je stav serveru že beží, kde čakáme kým príde na sever request o napojení sa klienta a ak príde tak sa podľa stavu zavolá príslušná metóda. Keď je tam stav kde sa čaká na prvého hráča , tak sa zavolá metóda na vytvorenie spojenia, ktorá ako argument berie ID hráča, teda predáme tomu v tomto stave ID player A a to podľa tohto ID vytvorí spojenie a inicializuje všetky potrebne premenne na fungovanie spojenia , teda napr. Hander na počúvanie a spracovanie dát od konkrétneho klienta, zároveň ho aj zapne ale k tomu neskôr, a CommunicationProtocol čo ide o komunikáciu medzi serverom a klientom, cez ktorý posiela server dáta klientovi. Podobne je to aj pre druhého hráča, ale ak už je stav reprezentujúci, že server je plný tak pri novom spojení sa spojenie na krátko nadviaže , server pošle udaj o tom že je plný a zatvorí toto spojenie(metóda určená na to aby zatvorila stream a aj klienta). Na strane klienta sa následne objaví informácia o tom.

Ak sa náhodou stratí spojenie s klientom, odchytáva sa to tak že búd nám poslanie cez stream vyhodí výnimku alebo posielame pravidelne flag reprezentujúci iba to ci stále mame spojenie s klientom. Ak detekuje alebo odchytíme tuto výnimku tak sa metódou, ktorá skúsi poslať obom hracom informáciu o tom, že sa ten druhy odpojil. Tato metóda je v try bloku, keďže nedetekujeme s ktorým sme stratili spojenie, tomu s ktorým sme nestratili príde flag o tom ,ukonči hru a da hráčovi vedieť, že sa ten druhy odpojil.

V druhom tásku beží metóda ServerGameLoop, kde nájdeme tak isto while loop dokým server beží. Ta ma za úlohu stále posielať už spomínane flagy na zisťovanie či mame stále spojenie. Ak obaja hráči sú už pripravený, teda došla na server sprava o tom, že sú pripravený tak prepošle jednorazovo všetky údaje pre začiatok hry. Teda načíta to hernú mapu zo súboru , podľa toho aká bolo zvolená a tu následne pošle, pošle to údaje o hráčovi, ako jeho staty a zároveň aj jeho priradene ID. Pošle to staty všetkých veži a príšer a tak isto to pošle flag, ktorý znamená že hra začala a klient ma už vykresľovať hru. V tomto bloku kódu sa vytvorí aj objekt hra na serveri, objekty hráčov a nastaví sa hodnota na true, ktorá reprezentuje ci hra už beží.

V tomto while loope nájdeme aj blok ktorý začne sa vykonávať až keď hra beží. V nej sa prvý krát zapnú stopwatche , ktoré stopujú herný čas. Stále keď ubehne určitý čas delta, ktorý je nastavený ako konštanta tak sa zavolá na hre metóda update, ktorá updajtuje hru a pošle sa obom hracom udaj o statoch hráčov, pretože mohli dostať nove peniaze alebo stratiť život.

Server poskytuje metódy, ktoré fungujú ako broadcast teda zavolaním metódy sa to pošle obom hráčom, jedna sa o všetko čo server bude posielať počas hry, teda napr. List kde mame renderovanie údaje o príšerách, akcie pre vezú a príšeru a atď..

**ClientHandler**

Objekt ktorý počúva a spracováva požiadavky poslane od klienta. Zavolaní metódy StartClient sa v novom tasku spusti while loop, ktorý bude bežať dokým server bude mat stav ON , v ktorom ak sú nejaké dáta prítomne v streame tak ich vytiahneme zistime typ packetu a podľa toho zavoláme príslušnú akciu, ktorá spracuje ďalšie dáta a nastaví údaje na serveri alebo v hre. Napr. to že klient pošle tower action tak sa na hre zavolá metóda na spracovanie tejto akcie.

**Player**

Trieda reprezentujúca hráča, teda v nej si pamätáme všetky údaje o hráčovi ako je život a hodnotu jeho peňazí, mame tu aj list v ktorom mame zoznam jeho veži a príšer, list príšer, ktoré už nežijú , a tak isto objekt MonsterSpawner. Tato trieda poskytuje metódy ako metódu na zistenie ci hráč žije, metódu na to aby sa hráčovi zobral život, metódy na pracovanie s jeho majetkom , teda jedna čo mu pridá peniaze a druha čo mu zoberie, ta vracia bool hodnotu na základe toho či mal dosť a odpísalo sa mu z majetku alebo sa vráti false ak nemal dosť peňazí, to využívame vtedy ak hráč chce urobiť akciu ,ktorá ho stoji peniaze. Ďalej sú tu metódy na pridanie veži a odobratie zo zoznamu veži a metóda na zabitie príšery, teda sa príšera pridá na zoznam mŕtvych príšer a posledná metóda ktorá vymaže všetky mŕtve príšery, to prejde list príšer a porovná objekty príšer a tie čo mame v liste mŕtvych z toho listu odoberie.

**Monster Spawner**

Ide o pomocný objekt pri oživovaní príšer. Môže sa stáť, že hráč pošle naraz viac príšer a to by spôsobilo, že by boli v hre hneď na sebe a prekrývali sa. Preto vznikla tato pomocná trieda, ak hráč pošle príšeru tak najprv sa pridá do fronty v tejto triede a ak v tejto fronte niekto čaká a prejde určitý čas (ten je meraný za pomoci objektu cooldown) tak to dovolí oživenie príšery do herného sveta.

Tato trieda ale potrebuje referenciu na list všetkých príšer, ktoré sú na mape, aby vedela do neho pridávať nové príšery tým ich oživiť a potrebuje triedu cooldown.

**Cooldown**

Objekt , ktorý dôkaze merať plynutie času , teda nastaví sa čas trvania a predaj sa tomu aj stopwatch od servera aby sme zbytočne ich nemali viacej.

Mame tu bool property , ktorá reprezentuje ci tento cooldown uplynul.

Tento objekt poskytuje metódu na zmenu času cooldawnu, to používame pri tom keď mame vežu, ta ma pre svoj útok cooldown a keď ju vylepšime bude mat menší cooldown.

Potom metódu start, ktorá nastaví, že cooldown ešte neuplynul a v novom tasku spusti metódu, ktorá beží dokým neuplynie daný čas, potom nastaví hodnotu na true a skonči sa tento task.

**Game Map**

Reprezentuje cely hernú mapu, teda pamätá si v dvojrozmernom poli celu mapu v ktorom mame uložene herne štvorce, Tile. Property ako výška a šírka mapy, lokácia kde sa oživuj príšery pre hráča A aj B.

Pri vytváraní tohto objektu sa ako argument predá dvojrozmerne pole ktoré sa už precitalo zo súboru a na základe neho to vytvorí hernú mapu.

Trieda poskytuje metódy ako ci je niektoré políčko prechodne čo vracia bool. To využívajú príšery počas svojho pohybu. Pýtajú sa mapy či je políčko vedľa nich prechodne a ak áno tak pôjdu cez neho. Ďalej metódu, ktorá zisťuje či na danej lokácii exituje políčko, metódu pre získanie

konkrétneho políčka a metódu na zmenu políčka na danom mieste nahradením za nove poskytnuté políčko.

**Game**

Objekt, ktorý ma v sebe uložene všetky herne entity ako sa hráči, herná mapa a konštanty, ktoré reprezentujú herne staty príšer, vežičiek, ale aj statov ktoré majú hráči na začiatku, alebo času na oživenie ďalšej príšery a atd.

Nájdeme tu aj list Factories ,ktoré nám vytvárajú konkrétne príšery a vežičky.

Nájdeme tu metódu GetTile., ktorá vráti políčko a metódy pre stavanie , vylepšovanie a ničenie veži. Teraz si uveďme príklad ak toto cele funguje. Na klientovi hráč vykoná danú akciu napr. Chce postaviť konkrétnu vežu. Tento udaj sa zabalí do štruktúry tower action a pošle sa to cez sieť. Client handler dostane tieto dáta a vykoná preto akciu, ktorá na objekte hra zavolá metódu action. Tato metóda si za pomoci metódy getTile vytiahne z mapy dane políčko na ktoré klikol klient a kde chce postaviť vežu. Na danom políčku sa zavolá metóda Action, Objekt políčko reprezentuje jeden štvorček mapy a reprezentuje neprechodný štvorček na ktorom neviem vykonať žiadnu akciu preto ma metódu action prázdnu. Od neho dedí potom políčko TranstitionalTile , ktoré reprezentuje políčko po ktorom chodia príšery a teda ma nastavenú property ktorá hovorí, že je prechodne. Potom mame políčko EmptyTowerTile ktoré dedí od Tile a pridáva tomu funkciu vlastníctva teda kto toto políčko vlastný, ktorý hráč. Tu už overridneme metódu action. Toto políčko nám reprezentuje prázdne políčko kde hráč vie postaviť vežu. A teda ak zavoláme na tomto políčko metódu action tak sa najprv overí, že či chceme vykonať akciu stavania, pre prípad, že by klient zabalil zlu tower action tak server musí mať vždy pravdu. Potom sa overí vlastníctvo či ma daný hráč, ktorý chce vykonať stavbu veží na tomto políčku vlastníctvo políčka a následne či ma hráč dosť peňazí na stavbu. Ak všetko je splnene postaví sa na danom mieste veža , teda sa zavolá metóda na postavenie veži a ktorá zmení na mape políčko za políčko reprezentujúce danú vežu za pomoci factory, pridá si do zoznamu veži a tuto vežu pošle cez server metódou,spať danú akciu ako potvrdenie pre klienta, že to vyšlo a môže vykresliť danú akciu ako, že sa stala a druhy hráč uvidí akú akciu spravil prvý hráč.

Takto podobne funguje aj pre ničenie vežičky a vylepšovanie.

Potom tu mame metódu UpdateGame , ktorá vykoná jeden krok hry a teda:

Zavolá metódu SpawnNewMonsters, ktorá pre oboch hráčoch skúsi oživiť novú príšeru, teda na objekte Monster Spawner skúsi zavolať metódu skúsi zavolať metódu, ktorá ak je cooldown načítaný tak oživí novu príšeru a znovu zapne cooldown alebo neoživí nič z dôvodu, že žiadna príšera nečaká alebo ešte efekt cooldownu stále neskončil.

Potom metódu MovePlayerMonsters pre oboch hráčov, ktorá pre všetky príšery daného hráča zavolá na objekte príšera metódu Move, ktorá pohne príšerou, zároveň sa tam detekuje či sa vie daná príšera pohnúť a to tak, že metóda Move vracia bool a ak vráti false, tak to znamená, že sa už nevie daná príšera pohnúť, pretože prišla nakoniec a tak sa daná príšera zabij , hráč dostane za ňu odmenu teda sa mu pripíšu peniaze a druhému hráčovi to zoberie život.

Metoda TowerMonsterClash, ktorá zavolá inú metódu, ale zavolajú dvakrát aby to bolo pre oboch hráčov, kde raz je hráč ako útočník a raz ako obranca. V tej ďalšej metóde sa pre každú vežu obrancu zavolá metóda na veži AttackToMonsterArmy, ktorá berie list príšer od útočníka. V tejto metóde sa už pozerá na konkrétnu vežu, kde sa pozerá či môže veža útočiť teda ci jej cooldown je na nule a ak áno tak sa prejde každá príšera zo listu a zisti sa ci je príšera v dosahu veže, ak áno zaútočí veža na príšeru a to tak, že sa zavolá metóda AttackToMonster, ktorá berie konkrétne jednu príšeru a na je

sa zavolá metóda TowerAttack kde ako argument sa predá daná veža. Vďaka dedičnosti tuto metódu mame pre každú konkrétnu vežu overridnutú.A vďaka tomu potom na objekte prizerá mame overloadnutú metódu pre každý typ veže a vieme spracovať pre každú príšeru ako ma reagovať na daný typ veže, či ju ignorovať ,alebo brat do úvahy , zobrať príšere život,alebo či ma daná príšera nejakú zotrvačnosť voči danej veže napr. že príšere menej berie Fire veža a teda sa jej zoberie o 30 percent menej života. Alebo ju veža spomalí.

Potom sa zavolá pre oboch hráčov metóda ktorá mŕtve príšery, ktoré už zomreli výmaze z hry.

Metoda Income, tam ide taktiež o cooldown ktorý je ako privátna premenná na objekte Game a ktorý ak už vyprchal tak obom hracom pridá peniaze a znovu zapne cooldown. To je na to aby hráči mali nejaký základný prijem peňazí a nedostali sa do pozícia, že nemajú peniaze na žiadnu akciu.

Potom sa pošle cez server renderovacia informácia o každej momentálnej žijúcej príšere aby klienti vedeli vykresliť tento stav hry.

A ako posledné sa zavolá dva krát, pre oboch hráčov metóda ktorá skontroluje ci náhodou nejaký hráč už nezomrel a ak áno tak to pošle informáciu o konca hry teda kto vyhral a kto prehral a ukonči to hru.

**Monsters a Towers**

Ide o objekty ktoré majú spoločného predka , konkrétne v, konkrétne veže majú predka Tower a konkrétne príšera majú predka Monster, tieto objekty už boli spomínane vyššie, ale v podstate ide o to že reprezentujú všetko čo majú vedieť príšery, teda možnosť sa pohnúť a podobne a všetko čo majú vedieť veže, útočiť a tak ďalej.