Martin Kvietok
ID: 127214

Data Modeling for a Turn-Based RPG Combat System

(Assignment 2 – Protocol)

LS 2024/2025

Contents

Návrh herného sveta a mechaník

Navrhovaný svet predstavuje online RPG hru s otvoreným svetom a ťahovým systémom, inšpirovanú klasickými hrami ako *Dungeons and Dragons*. Každý hráč (Character) si môže vytvoriť a nakonfigurovať vlastnú postavu vrátane výberu hernej triedy (Class) a základných atribútov ako sú **Sila (Strength), Obratnosť (Dexterity), Odolnosť (Constitution), Inteligencia (Intelligence)** a **Zdravie (Health)**.

Hra obsahuje jednoduchý súbojový (Combat) systém, do ktorého sa hráči môžu dynamicky pripájať. Súboj prebieha v ťahoch (Round), počas ktorých hráči útočia na protivníkov s cieľom ich poraziť a získať ich predmety. Mechanika útoku je založená na náhodnosti – využíva sa hod 20-stenovej kocky, pričom výsledok určuje, či útok zasiahol cieľ. Obrana protivníka závisí od hodnoty jeho brnenia, ak je vyššia než spôsobený útok, môže dôjsť k jeho zablokovaniu.

Každá postava má obmedzený počet **akčných bodov (Action Points)**, ktoré môže využiť na útoky alebo čarovanie. Okrem toho disponuje aj **inventárom** s obmedzenou kapacitou, kde si uchováva získané predmety.

V prípade porážky postava stratí svoje predmety, ktoré zostanú na bojisku a môžu byť získané inými hráčmi. Hráčova postava je následne znovu oživená (respawn) v hernom svete, pričom začína s prázdnym inventárom a základným vybavením.

Implementácia

Character Package

Character predstavuje základnú entitu hráča v hernom svete. Každý hráč je identifikovaný jedinečným identifikátorom (ID) a môže si nakonfigurovať vlastné atribúty podľa svojich preferencií. Pri tvorbe postavy je hráč obmedzený počiatočným počtom bodov (tzv. štartovací kapitál), ktoré si musí strategicky rozdeliť medzi dostupné atribúty.

Medzi základné atribúty patria:

- **Strength (sila)**
- **Dexterity (obratnosť)**
- **Constitution (odolnosť)**
- **Intelligence (inteligencia)**
- **Health (zdravie)**

Týchto päť atribútov je povinných pre každú postavu. Návrh systému však podporuje možnosť rozšírenia a pridávania ďalších atribútov v budúcnosti. Atribúty sú reprezentované samostatnou entitou **Attribute**, ktorá obsahuje informácie o type a vlastnostiach atribútu. Prostredníctvom relačnej entity **CharacterAttribute** sú priraďované konkrétnym postavám spolu s nastavenými hodnotami.

Každá postava si musí zvoliť aj **herné povolanie (Class)**, ktoré určuje štýl hrania a poskytuje špecifické bonusy. Každá trieda definuje jedinečné modifikátory ovplyvňujúce:

- Maximálny počet akčných bodov (**MaxAP**) = (Dexterity + Intelligence)ClassModifier
- Maximálne zdravie (**MaxHealth**) = Health
- Hodnotu brnenia (**Armor**) = 10 + (Dexterity /2) + ClassArmorBonus
- Veľkosť inventára (**MaxInventorySize**) = (Strength + Constitution)ClassInventoryModifier

V tabuľke **CharacterAttributes** sa hodnoty atribútov postavy vyhľadávajú podľa ID postavy. Tieto hodnoty sa následne vynásobia koeficientom priradeným k jej **Class** (povolaniu). Týmto spôsobom sa atribúty postavy prispôsobujú jej povolaniu a môžu sa dynamicky meniť v závislosti od vybranej triedy. A inicializujú sa bezprostredne po výbere povolania a počas hry sa ďalej vyvíjajú priamoúmerne podľa aktuálnych hodnôt pridelených v atribútoch.

Ďalšou vlastnosťou postavy je **Status**, ktorý vyjadruje jej aktuálny stav. Hráč môže byť buď:

- v **boji (In Combat)** – aktívne sa zúčastňuje súboja,
- **odpočívať (Resting)** – nachádza sa mimo boja a postupne regeneruje zdravie.
- **nerobiť nič (Idle) –** postava nevykonáva žiadnu akciu, iba sa nachádza v prostredí

Entita Character ďalej obsahuje dynamické atribúty, ktoré sa menia počas hrania, ako napríklad:

- **CurrentWeight** – aktuálna záťaž postavy (súvisí s inventárom),
- **CurrentHealth** – aktuálne množstvo zdravia,
- **CurrentAP** – aktuálny počet akčných bodov.

Tieto hodnoty sú dôležité najmä počas súbojov, kde sa priebežne aktualizujú v závislosti od vykonaných akcií a prijatých zranení.

SCENÁR 2:

Celý proces sa odohráva v rámci entity **Character** a jej atribútov.

Ak postava v hernom svete nemá záujem zúčastniť sa boja alebo bola vyčerpaná predchádzajúcim bojom, môže si aktualizovať svoj status na Resting. V tomto stave sa jej CurrentHealth postupne regeneruje podľa prednastavených časových intervalov (napríklad každú sekundu sa zvýši o 1 HP) Regenerácia CurrentHealth je neustále kontrolovaná v porovnaní s maximálnym počtom životov (MaxHealth), aby sa zabezpečilo, že hodnota CurrentHealth neprekročí túto maximálnu hranicu. Ak sa hodnota CurrentHealth rovná MaxHealth, regenerácia sa zastaví a postava prechádza do stavu Idle.

V prípade úmrtia v boji sa atribúty postavy CurrentHealth, CurrentWeight a CurrentAP resetujú do pôvodného stavu: CurrentHealth sa nastaví na hodnotu MaxHealth, CurrentWeight sa nastaví na hodnotu 0 a CurrentAP sa nastaví na hodnotu MaxAP. Tento reset zabezpečuje, že postava začne od základného stavu po obnovení alebo oživení.

Spell Package

Entita Spell reprezentuje všetky kúzla a útoky dostupné v hre. Každý spell (kúzlo alebo útok) je jedinečný a obsahuje nasledovné základné vlastnosti:

- **ID** – unikátny identifikátor,
- **Name** – názov kúzla,
- **BaseCost** – základná cena v akčných bodoch (koľko **AP** je potrebné na jeho aktiváciu),
- **BaseDamage** – základné poškodenie (koľko životov uberá cieľu)
- **Accuracy** – pravdepodobnosť úspešného vykonania (v rozsahu 1–20, pričom čím vyššia hodnota, tým presnejšie kúzlo)

Každý spell je priradený do jedinej kategórie (Category), ktorá ovplyvňuje jeho špecializáciu. Príkladmi kategórií sú napríklad ohnivá mágia, boj zblízka, ľadové kúzla a podobne. Kategória obsahuje vlastný **CategoryBaseCostModifier**, ktorý upravuje celkový náklad na použitie kúzla – napr. ohnivá mágia môže byť energeticky náročnejšia než fyzický útok.

Modularita cez SpellAttributeEffect

Každé kúzlo má priradené efekty atribútov (**SpellAttributeEffect**), ktoré sú nevyhnutné pre jeho výpočet a funkčnosť. Táto entita definuje, ako sú konkrétne atribúty postavy (**Attribute**) prepojené s daným kúzlom a aký vplyv majú na výsledné hodnoty (napr. náklady, poškodenie, liečenie).

Každý SpellAttributeEffect definuje:

- **Action** – aký aspekt kúzla ovplyvňuje (Damage, CostAP, Heal)
- **ModifierType** – spôsob aplikácie modifikácie (napr. **pripočítanie, odpočítanie, násobenie, delenie**).

Príklad: Ak kúzlo spôsobujúce poškodenie závisí od inteligencie a obratnosti postavy, a vo vzorci sa tieto atribúty pripočítavajú k základnému poškodeniu, kúzlo bude obsahovať dva záznamy v entite SpellAttributeEffect:

- jeden pre **Action = Damage**, **ModifierType = +**, **AttributeID = Intelligence_ID**,
- druhý pre **Action = Damage**, **ModifierType = +**, **AttributeID = Dexterity_ID**

Týmto spôsobom je možné implementovať modulárny a flexibilný systém kúziel, v ktorom každý spell môže byť ovplyvňovaný viacerými atribútmi hráča (napr. Inteligenciou, Silou, Odolnosťou) a zároveň môže mať rôzne účinky v závislosti od jeho konfigurácie. Tento návrh umožňuje vysokú mieru rozšíriteľnosti a variabilitu – vývojári môžu jednoducho pridávať nové kúzla, efekty a kategórie bez zásahu do základnej logiky systému.

Inventory Package

Balík **Inventory** reprezentuje systém správy predmetov v hre. Predmety sú reprezentované entitou **Item**, pričom každý predmet obsahuje nasledovné základné vlastnosti:

- **Name** – názov predmetu,
- **Weight** – hmotnosť,
- **Type** – typ predmetu (napr. zbraň, zbroj, artefakt, surovina),
- **ItemModifier** – špeciálne vlastnosti predmetu, ktoré zvyšujú efektivitu niektorých spellov alebo ovplyvňujú schopnosti postavy (týka sa najmä unikátnych predmetov a výbavy ako zbrane, talizmany a pod.).

Stavy predmetu

Každý predmet sa môže nachádzať v jednom z dvoch základných stavov:

1. V inventári postavy

Ak je predmet súčasťou inventára postavy, vytvára sa entita **InventoryItem**, ktorá reprezentuje väzbu medzi konkrétnym **Character** a predmetom (**Item**). Táto entita zároveň uchováva informáciu o tom, či je predmet aktuálne vybavený (**isEquipped = true**) alebo nie.

- Príklad: magická palica bude mať isEquipped = true, ak ju čarodejník aktívne používa.
- Naopak, vzácny zberateľský predmet môže byť v inventári, ale bez aktívneho použitia (isEquipped = false).

2. Na bojisku

Ak predmet nie je vlastnený žiadnou postavou, nachádza sa priamo na bojisku a je reprezentovaný entitou **BattlegroundItem**. Táto entita uchováva:

- Väzbu na konkrétny **Combat**, v ktorom sa predmet nachádza,
- Informáciu, či už bol predmet niekým vyzdvihnutý (**isTaken** = true/false).

Pokiaľ isTaken = false, môže si predmet vyzdvihnúť ktorýkoľvek účastník daného boja. Ak je však hodnota true, predmet je už priradený inému hráčovi a nie je viac dostupný na vyzdvihnutie.

SCENÁR 5:

Počas boja môže dôjsť k situácii, keď postava zomrie – teda jej aktuálna hodnota zdravia (**CurrentHealth**) klesne na nulu alebo menej. V takom prípade systém automaticky presúva všetky predmety, ktoré postava vlastnila, na bojisko. Tento proces prebieha tak, že v tabuľke **InventoryItem** sa záznamy o predmetoch priradených zosnulej postave odpoja (nastavením hodnoty Character_ID na null), stav predmetov sa zmení na BATTLEGROUNDITEM a zároveň sa pre každý takýto predmet vytvorí nový záznam v tabuľke **BattlegroundItem**, ktorý ho jednoznačne viaže na konkrétny boj (**Combat**).

Takto presunuté predmety sú následne považované za voľne dostupné a môžu byť počas ďalších kôl boja získané inými postavami. Predmety dostupné na bojisku možno získať vyfiltrovaním tabuľky **BattlegroundItem** podľa konkrétneho boja. Okrem predmetov, ktoré sa na bojisko dostali smrťou postavy, môžu byť na začiatku boja prítomné aj náhodne vygenerované predmety. Tie nepatria žiadnemu konkrétnemu charakteru a sú do boja vložené podľa definovaných pravidiel danej situácie či scenára.

SCENÁR 6:

Počas svojho ťahu má každý účastník boja (**CombatParticipant**) možnosť vykonať špeciálnu akciu namiesto útoku – konkrétne zdvihnúť (získať) predmet z bojiska. V takom prípade sa v systéme zaznamená udalosť do tabuľky **CombatLog**, pričom sa nastaví typ udalosti (**Event_Msg_Type**) na ***Pick Item***. Záznam zároveň identifikuje hráča, ktorý predmet získal – teda na strane útočníka sa uloží jeho **CombatParticipant_ID**. Zároveň sa do logu uvedie identifikátor zvoleného predmetu (**BattlegroundItem_ID**), ktorý hráč chcel získať.

Následne systém z predmetu zistí jeho hmotnosť (**Weight**) a pokúsi sa ju pripočítať k aktuálnej nosnosti hráča (**CurrentWeight**). Ak by však táto nová hodnota prekročila maximálnu povolenú hmotnosť (**MaxInventoryWeight**) daného charakteru, akcia zlyhá a v **CombatLog** sa nastaví **MoveSucceeded** na hodnotu *false*. Predmet sa v takom prípade hráčovi nepriradí.
Ak však nosnosť hráča umožňuje predmet získať, operácia pokračuje nasledovne: v tabuľke **BattlegroundItem** sa danému predmetu nastaví príznak **IsTaken** na *true*, čím sa zabezpečí, že predmet už nebude dostupný pre iných hráčov. Pomocou **Item_ID** sa následne v tabuľke **Item** aktualizuje stav predmetu na *INVENTORYITEM*. V tabuľke **InventoryItem** sa buď vytvorí nový záznam, alebo sa aktualizuje existujúci záznam pre daného hráča (na základe **Character_ID** a **Item_ID**), čím sa predmet priradí do jeho inventára.
Ak je získaný predmet zbraň, hráč si ju môže rovno nastaviť ako vybavenú (**IsEquipped** = true), a tým ju začať používať počas nasledujúcich bojových akcií.

Combat Package

Balík **Combat** reprezentuje mechaniku bojov, ktoré prebiehajú medzi viacerými postavami (**Character**) v rámci definovaných kôl. Každý **boj (Combat)** je jednoznačne identifikovaný pomocou ID a pozostáva z viacerých **kôl (Round)** , ktorých počet závisí od počtu účastníkov a priebehu samotného boja.

Štruktúra kola

Každé kolo v rámci boja obsahuje:

- **Poradové číslo kola**,
- **Čas začiatku a konca kola**, čo umožňuje systému v reálnom čase určiť, ktoré kolo práve prebieha. Ak sa nová postava (Character) pripája do prebiehajúceho boja, zaradí sa vždy na koniec aktuálnej rady.

Účasť postavy v boji

Zapojenie postavy do boja sa uskutočňuje vytvorením inštancie entity **CombatParticipant**, ktorá reprezentuje konkrétneho účastníka v danom súboji. V tomto procese sa zaznamenáva presný čas vstupu postavy do boja (**JoinTime**) a zároveň sa sprístupňujú všetky aktuálne informácie o stave postavy – ako napríklad **CurrentHealth**, **CurrentAP** a **CurrentWeight** – prostredníctvom väzby na entitu **Character** (foreign key). Systém zároveň zabezpečuje kontrolu, či je postava stále nažive a či už nie je súčasťou daného boja, čím sa predchádza duplicitnému zapojeniu tej istej postavy do jedného súboja.

SCENÁR 3:

Celý proces sa odohráva v rámci vzťahu medzi entitami **Character** a **Combat**, pričom pre každý **Combat**, v ktorom sa postava chce zúčastniť, sa vytvorí samostatný záznam v tabuľke **CombatParticipant**. Tento záznam obsahuje hodnotu **JoinTime**, ktorá sa nastaví na aktuálny čas, a **IsAlive**, ktorá je nastavená na **True**. Postava sa následne pripojí do aktuálneho kola. V tabuľke **Round** sa porovnáva, či je **JoinTime** v rozpätí medzi alebo pred **StartTime** a **EndTime** daného kola a či je hodnota **IsAlive** nastavená na **True**. Týmto spôsobom sa zabezpečí, že postava sa zapojí do správneho kola boja, ak je stále nažive. Postavy sú triedené vzostupne podľa **JoinTime**, takže tá, ktorá sa pripojila ako posledná, vykoná svoju akciu ako posledná.

Záznam akcií počas boja

Počas každého kola sa zaznamenávajú všetky akcie vykonané účastníkmi boja ako napríklad: použitie kúzla, zdvihnutie predmetu, smrť postavy**.** Tieto akcie sú uchovávané v entite **CombatLog**, pričom každý záznam obsahuje:

- Typ udalosti (**Event_Msg_Type**) – útok, smrť, získanie predmetu
- Príslušné **ID boja a kola**, v ktorom akcia prebehla
- Presný **čas vykonania akcie**

V prípade útoku sa hádže 20-stennou kockou a výsledok sa zaznamená do atribútu **DiceThrow** (v ostatných prípadoch je hodnota null). Ďalej sa eviduje:

- Kto kúzlo použil (útočník) a na koho bolo nasmerované (cieľ) – pomocou **ID útočiaceho a cieľového CombatParticipant**,
- **ID použitého kúzla**,
- Na základe týchto údajov sa počítajú odvodené atribúty ako
    - **APCost = BaseCostCategory.Modifier (1 − (SelectedAttribute/100))(1 − ItemModifiers)**
    - **Damage = BaseDamage(1 + ConfiguredAttribute/20)**

Ak bol útok úspešný, atribút **MoveSucceeded** sa nastaví na hodnotu true a aktualizujú sa príslušné štatistiky účastníkov boja, ktorí boli akciou ovplyvnení.

SCENÁR 1 + 4:

V rámci tohto procesu sa vytvára záznam v **CombatLog**, kde sa nastaví **Event_Msg_Type** na hodnotu "Cast Spell". V tomto prípade sa nastaví aj atribút **ID útočiaceho** ako kľúč na zodpovedajúci záznam v **CombatParticipant**. Útočiaca postava si následne vyberie **spell**, ktorý chce zoslať, a tento **spell** sa priradí k príslušnému záznamu. V tabuľke sa nastaví **ID tohto spellu** a tiež sa určí, na koho tento spell bude použitý. V prípade, že ide o **heal**, môže byť cieľom aj samotná postava (self-target).

Teraz dochádza k hodu kockou, pri ktorom systém vygeneruje náhodné číslo od 1 do 20 a uloží ho do tabuľky **DiceThrow**. Táto hodnota slúži na určenie úspešnosti kúzla – porovnáva sa s atribútom **Accuracy** z entity **Spell**. Ak je hod kockou menší alebo rovný ako **Accuracy**, kúzlo zasiahne cieľ. V opačnom prípade kúzlo minie a premenná **MoveSuccess** sa nastaví na hodnotu false.

Následne ak útočník zasiahol cieľ sa začne vypočítavať **APcost** a **Damage** podľa vzorca.

AP Cost:

1. Najprv si v tabuľke **Spell** podľa ID nájdeme príslušné kúzlo.
2. Následne v tabuľke **SpellAttributeEffect** vyfiltrujeme všetky záznamy, kde je hodnota **Action** nastavená na **CostAP**.
3. Pre každý z týchto efektov dohľadáme atribúty, ktoré sú namapované na daného **Charactera**.
4. Z tabuľky **CharacterAttributes** získame aktuálne hodnoty týchto atribútov a použijeme ich vo vzorci.
5. Zohľadníme aj typ modifikácie (**ModifierType**), ktorý je prednastavený ako **"+"** – teda hodnota sa pripočítava.
6. Skontrolujeme tabuľku **InventoryItems**, kde podľa ID postavy overíme, či má postava nejaký predmet so stavom **isEquipped = True**. Ak áno, jeho efekt sa taktiež zohľadní vo výpočte.

Damage:

- Rovnaký proces ako pri výpočte **AP cost**, ale v tabuľke **SpellAttributeEffect** filtrujeme iba tie záznamy, kde je **Action** nastavená na **Damage**.

Heal:

- Výpočet **healu** prebieha identicky ako pri výpočte **Damage**, len s tým rozdielom, že **Action** je nastavená na **Heal**.

Následne sa porovná hodnotu **APCost** so **súčasným CurrentAP útočníka**, ktorého nájdeme podľa jeho ID v tabuľke **CombatParticipant**. Ak má postava **dostatok akčných bodov**, potrebná hodnota sa **odpočíta** z CurrentAP. Ak nie spell sa nevykonná a **MoveSuccess** sa nastaví na hodnotu false.

Následne sa **damage** porovná sa s hodnotou **Armor** cieľovej postavy.

- Ak je damage **menší než armor**, útok sa považuje za neúspešný a **MoveSucceeded = false**.
- V opačnom prípade sa **MoveSucceeded = true** a vypočítaný damage sa **odpočíta z CurrentHealth cieľa**.

Tento proces sa **opakuje, kým sa neminú všetky AP** aktuálnej postavy. Následne sa na rad dostáva ďalšia postava.

Úmrtie postavy

Ak hodnota CurrentHealth postavy **klesne na 0 alebo menej**, postava **zomiera**. V tom prípade:

- Všetky jej predmety sa označia ako **bojové predmety (BattlegroundItem)**.
- V tabuľke InventoryItem sa jej predmety nastavia na **null**, čím sa uvoľnia z inventára. (ITEM_ID == null)

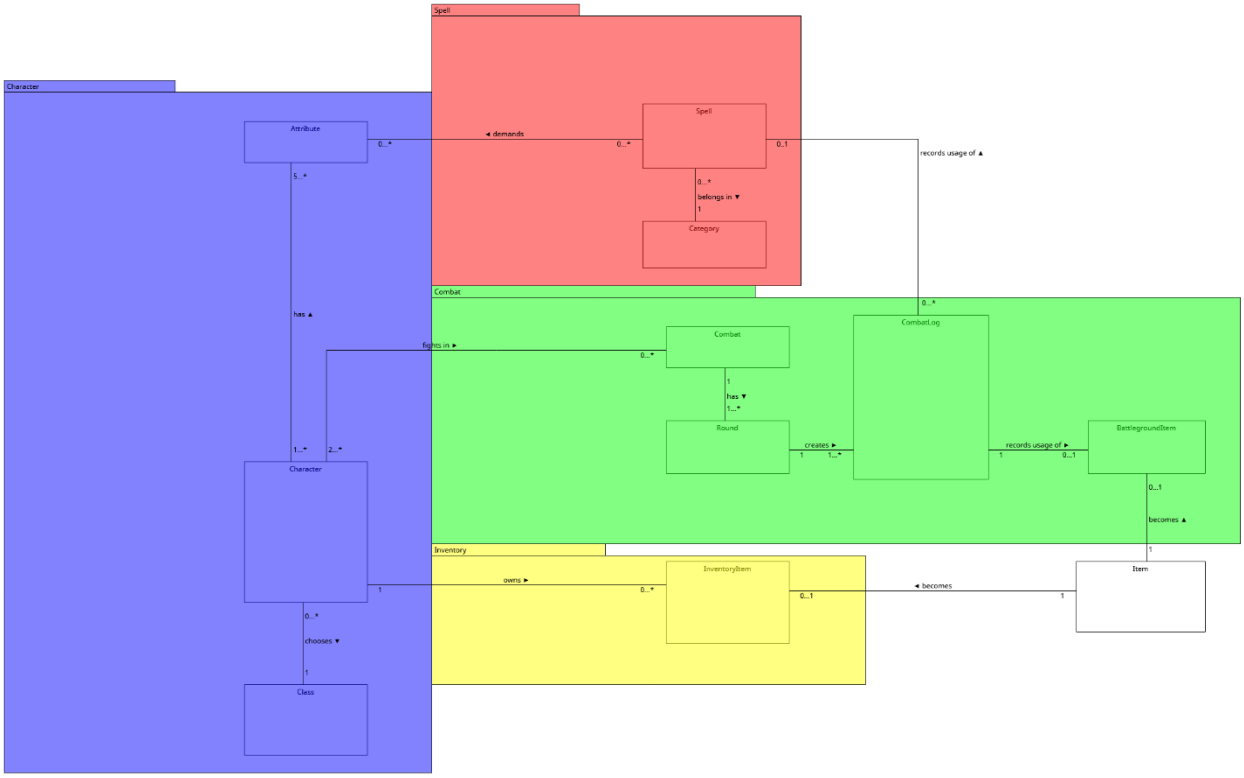
Na **konci kola** sa každému účastníkovi, ktorý má **IsAlive = true**, nastaví hodnota CurrentAP späť na maximum, teda sa **skopíruje MaxAP → CurrentAP**.

Interakcia s predmetmi

V prípade, že sa postava rozhodne zdvihnúť predmet z bojiska, zaznamenáva sa **ID predmetu**, ktorý si vyzdvihla. Tento záznam je taktiež súčasťou bojového logu. Tento návrh systému boja poskytuje robustný a rozšíriteľný rámec, ktorý umožňuje sledovať všetky podstatné udalosti v rámci každej fázy boja a zabezpečuje konzistentnosť údajov o postavách, kúzlach a ich interakciách.

Prílohy

E-R diagram



Martin Kvietok
ID: 127214

Logický model