Návrh testovací strategie

Popis funkcionality aplikace

**Dungeon Explorer je JavaFX aplikace pro prozkoumávání generovaného dungeonu.
Umožňuje:

- Spuštění nové hry nebo načtení uložené hry
- Pohyb postavy po místnosti se zpracováním kolizí
- Zobrazení přehledu mapy a inventáře jako překrývky
- Interakci s dveřmi (přechod mezi místnostmi), sběr předmětů a otevírání truhlic
- Souboje s nepřáteli s různými útoky a cooldowny
- Ukládání a načítání stavu hry (mapa, inventář, vybavení)
- Přepínání úrovně logování a pauzovací menu s možností uložení nebo návratu do hlavního menu

Přehled částí aplikace

- Hlavní menu (Nová hra, Načíst hru, Přepnutí logování)
- Herní obrazovka s výhledem na dungeon
- Pohyb a kolize postavy
- Přehled mapy (overlay)
- Inventář a výbava (overlay)
- Interakce s objekty (dveře, truhly, předměty, nepřátelé)
- Soubojový režim s útoky a cooldowny
- Ukládání a načítání stavu hry (Pause menu)

Prioritizace částí aplikace

**Kritéria:

- Frekvence použití: Jak často hráči danou funkci využívají.
- Dopad na pokračování ve hře: Selhání znemožní postup nebo způsobí ztrátu postupu.
- Pravděpodobnost selhání: Technická náročnost nebo riziko chyb.

**Návrh prioritizace*
Priorita:

- A – vysoká

- B – střední
- C – nižší

Část aplikace	Priorita	Odůvodnění
Hlavní menu (Nová hra, Načíst hru)	A	Základní vstup do hry; bez něj nelze začít ani pokračovat.
Generování mapy	A	Bez generování dungeonu není obsah místností; kritické pro samotnou existenci hry.
Pohyb a kolize postavy	A	Klíčové pro hratelnost—bez pohybu a správné detekce kolizí hra ztrácí smysl.
Interakce s dveřmi (přechod mezi místnostmi)	A	Umožňuje postup hrou; selhání blokuje další průchod.
Soubojový režim	A	Jádro herní mechaniky; bez něj není žádná výzva ani odměna.
Ukládání a načítání hry (Pause menu)	A	Zajišťuje zachování postupu; chyby vedou ke ztrátě času a frustraci hráče.
Interakce s předměty a truhlicemi	B	Důležité pro zisk vybavení a motivaci, ale hráč může bez nich pokračovat.
Inventář a výbava (overlay)	B	Často využívané pro správu předmětů, střední dopad na plynulost hry.
Přehled mapy (overlay)	B	Pomáhá orientaci, ale není nezbytně nutný pro základní průchod hrou.
Renderování místnosti (RoomRenderer)	B	Zajišťuje vizuální stránku dungeonu; chyby ovlivní zážitek, ale ne vždy zcela zablokují hru.
Přepínání logování	C	Nástroj pro vývoj a diagnostiku; hráči ani herní logice nebrání.

Test Levels

- **Revize** – odhalení chyb v požadavcích a návrhu dřív, než se začne kódovat
- **Vývojářské testy** – unit testy s co největším pokrytím, včetně hraničních hodnot
- **Integrační testy** – zaměření na kritické vazby mezi moduly
- **Systémové testy** – end-to-end testy simulující běžného hráče
- **UAT** – ověření reálnými uživateli, že aplikace splňuje požadavky
- **Testování v produkci** – základní kontrola po nasazení

Intenzita

- **ano** – povinná kontrola/test

- **vysoká** – plné pokrytí hlavních scénářů, regresní testy, okrajové případy
- **střední** – hlavní scénáře, běžné situace
- **nízká** – základní funkčnost, smoke test

Návrh test levels

Část aplikace	Revize	Vývojářské testy	Integrační testy	Systémové testy	UAT	Testování v produkci
Hlavní menu (Nová hra, Načíst hru)	ano	vysoká	vysoká	vysoká	vysoká	ano
Generování mapy	ano	vysoká	vysoká	vysoká	vysoká	ano
Pohyb a kolize postavy	ano	vysoká	vysoká	vysoká	vysoká	ano
Interakce s dveřmi (přechod mezi místnostmi)	ano	vysoká	vysoká	střední	střední	ano
Soubojový režim	ano	vysoká	vysoká	vysoká	vysoká	ano
Ukládání a načítání hry (Pause menu)	ano	vysoká	vysoká	vysoká	vysoká	ano
Interakce s předměty a truhlicemi	ano	střední	střední	střední	střední	ano
Inventář a výbava (overlay)	ano	střední	střední	střední	střední	ano
Přehled mapy (overlay)	ano	střední	střední	střední	střední	ano
Renderování místnosti (RoomRenderer)	ano	střední	střední	střední	střední	ano
Přepínání logování	ano	nízká	nízká	nízká	nízká	nízká

Detailní testovací scénář

DTS: Otevření truhly pomocí klíče – úspěšný scénář

Popis:
Hráč sebere klíč ležící v místnosti, poté otevře truhlu a získá náhodný loot.

Cíl testu:
Ověřit, že:

1. Stisknutí interakční klávesy („E“) u klíče jej přidá do inventáře a odstraní z místnosti.
2. Otevření truhly spotřebuje klíč, truhla zmizí a hráč získá položku z loot poolu.

Oblast systému:
RoomController.pickupItem, RoomController.openChest, InventoryController, InteractionService

Požadavky:

- V místnosti je alespoň jeden objekt typu `Item` s typem `"key"`.
- V místnosti je truhla (`Chest`) definovaná v `chests.json`, která vyžaduje klíč.
- Loot pool v definici truhly není prázdný.

Milníky:

- Klíč se objeví v inventáři a zmizí z místnosti.
- Po otevření truhly inventář ztrácí klíč a přibývá jeden nový předmět z loot poolu.
- Truhla už není vykreslena v místnosti.

Kroky testu:

1. Spustit novou hru a zobrazit počáteční místnost, ve které jsou klíč a truhla navzájem dosažitelné.
2. Použít pohybové klávesy (W/A/S/D) k přesunu postavy přímo vedle klíče.
3. Stisknout klávesu **E** pro interakci s klíčem.
4. Otevřít inventář (klávesa **I**) a ověřit, že se v seznamu objevil položka `"key"`.
5. Zavřít inventář a pohybovat se k truhle.
6. Stisknout **E** pro interakci s truhlou.
7. Znovu otevřít inventář a ověřit, že:
   - Klíč (`"key"`) již není v inventáři.
   - V inventáři je nová položka odpovídající některému ID z loot poolu (např. `"bronzeArmor"` nebo `"ironDagger"`).
8. Zkontrolovat, že v herní místnosti již není vykreslena truhla.

Očekávaný výsledek:

- Po kroku 3 klíč zmizí z herní plochy a objeví se v inventáři.
- Po kroku 6 se klíč odstraní z inventáře, truhla zmizí z místnosti, v inventáři je jedna nová položka.
- Žádné chybové hlášky se neobjeví, všechny UI prvky reagují správně.

Odhad pracnosti:

- Návrh: 10 min
- Vykonání: 15 min

Test vstupů

Testy vstupů – parametry generátoru mapy
ID Testu: MAP1
Popis: Pairwise testování validace tří vstupních parametrů třídy `MapGenerator`: `gridSize`, `minRooms`, `maxRooms`.

Parametry a třídy ekvivalence

- **gridSize**
  - **EC1 (platné):** celé číslo ≥ 3 a ≤ 100
  - **EC2 (neplatné):** celé číslo < 3
  - **EC3 (neplatné):** celé číslo > 100
- **minRooms**
  - **EC1 (platné):** 1 ≤ minRooms ≤ gridSize²
  - **EC2 (neplatné):** minRooms < 1
  - **EC3 (neplatné):** minRooms > gridSize²
- **maxRooms**
  - **EC1 (platné):** minRooms ≤ maxRooms ≤ gridSize²
  - **EC2 (neplatné):** maxRooms < minRooms
  - **EC3 (neplatné):** maxRooms > gridSize²

Očekávané výsledky

- **Platná kombinace (EC1, EC1, EC1):** konstruktor `MapGenerator(gridSize, minRooms, maxRooms)` proběhne bez výjimky.
- **Jakkoli neplatná kombinace (EC2 nebo EC3 v libovolném parametru):** konstruktor vyhodí validační výjimku (např. `IllegalArgumentException`).

Pairwise testovací případy

#	gridSize	minRooms	maxRooms	Očekávaný výsledek
TC1	10	1	5	úspěšná konstrukce (EC1, EC1, EC1)
TC2	2	0	5	výjimka (EC2 gridSize, EC2 minRooms)
TC3	200	200	5	výjimka (EC3 gridSize)
TC4	200	0	3	výjimka (EC3 gridSize, EC2 minRooms)
TC5	2	1	3	výjimka (EC2 gridSize)
TC6	10	200	3	výjimka (EC3 minRooms)
TC7	10	0	200	výjimka (EC2 minRooms, EC3 maxRooms)
TC8	2	200	200	výjimka (EC2 gridSize, EC3 minRooms/maxRooms)
TC9	200	1	200	výjimka (EC3 gridSize, EC3 maxRooms)

**Generoáno pomocí AllPairs:

```
from allpairspy import AllPairs

if __name__ == "__main__":
    params = {
        'gridSize': [10, 2, 200],
        'minRooms': [1, 0, 200],
        'maxRooms': [5, 3, 200],
    }

    for combination in AllPairs(list(params.values())):
        case = dict(zip(params.keys(), combination))
        print(case)
```

Test průchodu

Proces: Otevření truhly
Význam: Umožnit hráči získat loot z truhly za použití klíče.

Tabulka stavů a přechodů s TDL 2

Uzel	Aktivita	Vstupní akce	Výstupní akce	Dvojice TDL 2
A	Vstup do místnosti	Start hry	Herní místnost s klíčem a truhlou se zobrazí	A → B → C
B	Pohyb k pozici klíče	Pohyb na souřadnice klíče	Hráč je „u klíče“	B → C → D B → C → E
C	Seber klíč	Stisk E u klíče	Klíč odstraněn z místnosti, přidán do inventáře	C → D → F C → E → C
D	Pohyb k truhle	Pohyb na souřadnice truhly	Hráč je „u truhly“	D → E → F
E	Otevři truhlu	Stisk E u truhly (klíč v inv.)	Truhla odstraněna, klíč spotřebován, loot přidán	E → F → — E → C → E
F	Konec procesu	Loot přidán do inventáře	Proces končí, loot je v inventáři	—

Dvojice pokrytí TDL 2

Pro pokrytí všech dvojic po sobě jdoucích přechodů (cesty délky 2 hran) je třeba zahrnout tyto sekvence:

1. A → B → C
2. B → C → D
3. B → C → E
4. C → D → E
5. C → E → C
6. D → E → F
7. E → F → —
8. E → C → E

Test průchodu (TP1)

Kategorie: Hlavní proces „Otevření truhly“
Cíl: Ověřit, že posloupnosti stavů A→B→C→D→E→F probíhají bez chyb a pokrývají všechny

dvojice TDL 2.
Oblast: Interakce s objekty, inventář, ověřování vstupů

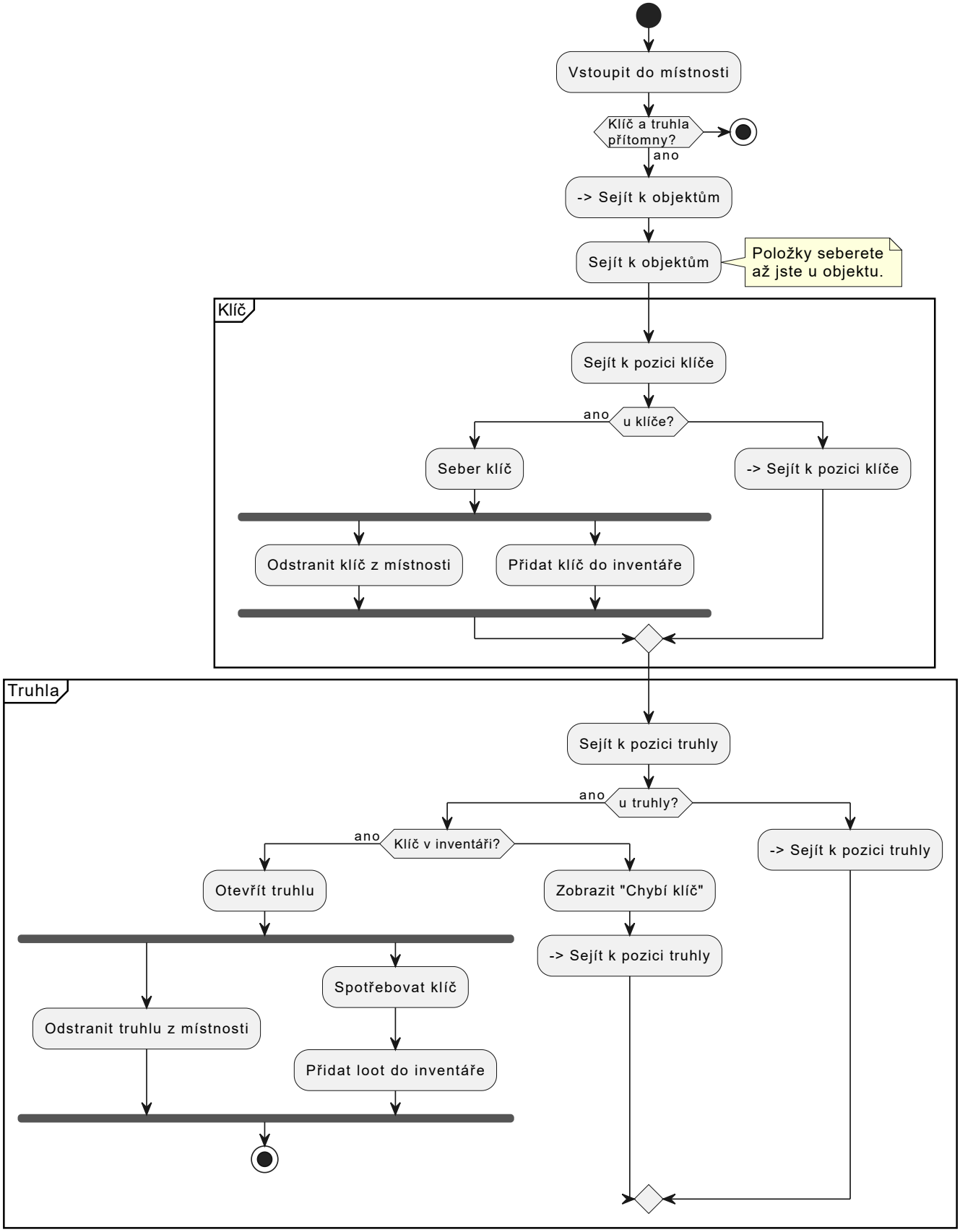
Kroky:

1. **(A)** Spustit hru → zobrazí se místnost s klíčem a truhlou.
2. **(B)** Pohyb k pozici klíče → hráč dosáhne klíče.
3. **(C)** Stisk E → klíč se sebere do inventáře.
4. **(D)** Pohyb k pozici truhly → hráč dosáhne truhly.
5. **(E)** Stisk E → truhla se otevře, klíč se spotřebuje, loot se přidá do inventáře.
6. **(F)** Otevřít inventář (např. stisk I nebo kontrola obsahu) → loot je viditelný.

Očekávaný výsledek:

- Všechny kroky proběhnou bez výjimek a chybových hlášek.
- Stavy a přechody odpovídají uzlům A…F.
- Test pokrývá všechny definované dvojice TDL 2.

Procesní diagram (PlantUML)


Vstoupit do místnosti
Klíč a truhla přítomny?
ano
-> Sejít k objektům
Sejít k objektům
Položky seberete až jste u objektu.
Klíč
Sejít k pozici klíče
ano
u klíče?
Seber klíč
-> Sejít k pozici klíče
Odstranit klíč z místnosti
Přidat klíč do inventáře
Truhla
Sejít k pozici truhly
ano
u truhly?
ano
Klíč v inventáři?
-> Sejít k pozici truhly
Otevřít truhlu
Zobrazit "Chybí klíč"
-> Sejít k pozici truhly
Spotřebovat klíč
Odstranit truhlu z místnosti
Přidat loot do inventáře