

Projekt z objektovo orientovaného programovania LS2025

Zámer projektu

Hráč v tejto Tower Defence hre ovláda našu planétu Zem a bráni ju pred neustálymi nájazdmi asteroidov, vesmírneho odpadu a ďalších nebezpečných hrozieb, ktoré pochádzajú z neživého (alebo možno nie tak úplne) vesmíru. Každý nepriateľský objekt predstavuje novú výzvu, a preto musí hráč neustále prispôsobovať svoju obrannú stratégiu aktuálnej situácii.

Stretnutie s nepriateľmi môže nielen vážne poškodiť samotnú planétu, ale aj spôsobiť náhodné negatívne efekty, ktoré ovplyvňujú priebeh hry. Tieto efekty môžu dočasne obmedziť schopnosti hráča rôznymi spôsobmi, napríklad znížením rýchlosti útoku, znížením spôsobeného poškodenia alebo dokonca úplným zablokovaním používania zbraní na určitý čas. Takéto situácie si vyžadujú premyslenú obrannú taktiku, aby sa hráč mohol účinne brániť a zároveň minimalizovať škody. Na druhej strane si hráč môže v obchode zakúpiť širokú škálu rôznych vylepšení a technologických pokrokov. Tieto vylepšenia dokážu posilniť obranu viacerými spôsobmi – môžu zvýšiť maximálny život, obnovovať poškodené zdravie atď.

Rozmanité typy nepriateľov si vyžadujú vysokú mieru flexibility pri výbere vylepšení a precízne plánovanie ich poradia, aby hráč mohol efektívne čeliť rôznym druhom útokov. Postupne rastúca obtiažnosť jednotlivých vln núti hráča neustále sa prispôsobovať novým výzvam, experimentovať s rôznymi hernými prístupmi a hľadať optimálne riešenia, čím sa každé hranie stáva jedinečným a dynamickým zážitkom. Hráč nebude mať za úlohu iba neustále odrážať nepriateľské útoky, ale zároveň bude musieť efektívne hospodáriť s dostupnými vylepšeniami, investovať do vývoja a objavovať nové, čoraz pokročilejšie efekty. Táto kombinácia akčného boja s premyslenou stratégiou vytvára dynamický a napínavý herný zážitok, ktorý ponúka množstvo možností na experimentovanie a postupné zdokonaľovanie herných zručností

Žáner: Tower defence

[illegible]

1. *LibGDX* – základná knižnica pre gui a prácu z opengl kontextom
2. *JUnit* – jednotkové testovanie
3. *Mockito* – pre overenie niektorých vecí bez ich úplného nastavenia
4. *Bytebuddy* – tak isto ako aj 3.

Ku kazdej podmienke napiste aspon jeden priklad kde v projekte je splnena, dodrzujte prosim poradie ako je na uvedene na stranke projekt:

- *Obsahovať dedenie a rozhrania:*
 - *Príklad: balík effects, kde máme abstraktnú triedu Effect a zopár rôznych efektov, každý zo ktorých dajakým spôsobom modifikuje svojho cieľa*
 - *Projek obsahuje celý balík interfaces z rôznymi použitými rozhraniami ako Mortal pre všetkých Entity, ktoré môžu zomrieť.*

- *Použitie základných OOP princípov*
 - *Emkapsulacia: v triede WaveCollection logiku pridania novej vlny a ohodnotenie nového setu vln enkapsulujem v metode addWave*
 - *Dedenie: už som povedal*
 - *Polymorfizmus: v triede GameScreen mam kolekcie projectileEnvironment, ktorá môže mať v sebe aj PlayerProjectile, aj EnemyProjectile*
 - *Abstrakcia: v Playeri mam pole RangedAttackingStrategy, ktorá obsahuje stratégiu útoku, ktorá “neviem ako funguje”, ale ja môžem použiť metodu attack playera, a ona už bude pracovať s tou stratégiou*
- *Musi obsahovať dostatok komentárov a anotácií: project obsahuje Javadoc a všetky metódy a polia sú podpísané tak, že sa sa maximálne ľahko pochopí, na čo oni slúžia*

- Musi byť jednotkovo otestovány: otestovano všetko okrem balik screens a tried MyGame a SpriteExtended, lebo oni potrebujú používať context opengl a aj tak sú gui

Element ^	Class, %	Metho...	Line, %	Branch...
✓ [] sk.stuba.fiit	95% (45/...	92% (23...	92% (56...	71% (96...
> [] effects	100% (6/6)	97% (38/...	98% (93...	60% (6/...
> [] entities	100% (3/3)	96% (49...	97% (10...	79% (27...
> [] enums	100% (2/2)	100% (4/4)	100% (1...	100% (0...
> [] events	100% (2/2)	100% (5/5)	100% (7...	100% (4...
> [] exceptions	100% (1/1)	100% (2/2)	100% (3...	100% (0...
> [] factories	100% (12...	100% (3...	100% (1...	50% (1/2)
> [] interfaces	100% (0/0)	100% (0/0)	100% (0...	100% (0...
> [] projectiles	100% (3/3)	100% (2...	100% (5...	71% (10/...
> [] strategies	100% (5/5)	77% (7/9)	88% (15...	100% (2...
> [] waves	100% (4/4)	100% (9/9)	100% (3...	100% (6...
© Collider	100% (1/1)	88% (8/9)	95% (22...	65% (13...
© EffectHandler	100% (1/1)	100% (7/7)	100% (2...	91% (11/...
© EffectUpgrade	100% (1/1)	100% (8/8)	100% (1...	50% (1/2)
© GameObject	100% (1/1)	90% (20...	94% (32...	50% (1/2)
© MyGame	0% (0/2)	0% (0/10)	0% (0/37)	0% (0/8)
© Timer	100% (1/1)	100% (11...	100% (2...	75% (6/8)
© WaveCollection	100% (1/1)	100% (3/3)	100% (2...	80% (8/...
© Weapon	100% (1/1)	100% (3/3)	100% (6...	100% (0...

- Musi splnať základné princípy softverového vývoja, vývoja objektovo orientovaného kódu a vývoja v jazyku Java 21: ono splna
- Použitie návrhových/architektonických vzorov
 - Strategy: Trieda Player obsahuje stratégie útoku, ktoré sa môžu dynamicky meniť (napr efekty)
 - Composite: EnemyProjectile má v sebe komponenty pre spracovávanie logicky odlišujúcich sa činností (kolízie – Collider, efekty - EffectHandler)
 - Decorator: ResistanceTakingDamageStrategy obsahuje poleTakingDamageStrategyWrapped, ktorej metóda takeDamage sa použije, keď to bude treba v ResistanceTakingDamageStrategy

- *Factory: v projekte je cely balik roznych factory pre vytvorenie zbrani, projectileov, spawnerov*
- *Logovanie zakladnych cinnosti: GameScreen vypisuje, ze hra je inicializovana, vlny informuju, ze boli vytvorene spawneri, MyGame vypisuje chybu pri inicializacii hraca z json*
- *Implementacia a pouzitie vlastnych vynimiek: ked hrac alebo nejaky super bol inicializovany nespravne (napr. health>maxHealth), bude vyhodena vynimka*
- *Implementacia a vytvorenie GUI: samotna hra uz je gui, rozne screeny, labely, tlacidla atd*
- *Explicitne pouzitie viacvlaknovosti: GameScreen vytvara nove vlakno pre inicializaciu novej vlny a retract starej*
- *Pouzitie generickovsti vo vlasnych triedach: rozhranie Factory<T> a triedy WeaponFactory impl Factory<Weapon>, voci comu musi implementovat metodu Weapon create(Object... args)*
- *Pouzitie reflexie: napríklad logovanie s pouzitim nazvy anon. triedy a vela pouzitia v testoch aby zistit zmeny vo flag-variables anon tried, vytvorených pre testovanie metod abstraktných tried*
- *Pouzitie lambda vyrazov, referencii na metody: assertThrows/assertNotThrow vo viacerych testoch, vytvoreni novych Runnable v EffectUpgrade atd*
- *Pouzitie serializacie/IO: Serializacia a deserializacia hraca*

Navod ako spustit project

Spustit main v projekt-DKraiser/lwjgl3/src/main/java/sk/stuba/fiit/lwjgl3/Lwjgl3.java