Minesweeper

- Dokumentace -

K. Bém, P. Šturz, L. Pražáková

Architektura

Řešení je rozděleno do několika souborů. Toto dělení je hlavně kvůli možnost lepšího rozdělení práce mezi více lidí a snížení možnosti konfliktů v gitu. Soubory:

- **common.py** - společné věci pro více tříd v různých souborech, takže nejvíce konstanty (rozměry, barvy), ale jsou zde i dvě funkce pro nastavení ikony hry a nastavení loga, ty se používají na obou obrazovkách
- **game.py** - přes tento soubor se hra spouští, je to obálka, která řídí přepínání mezi jednotlivými okny a také spouští hudbu na pozadí
- **gamestate.py** - v tomto souboru je business logika hry, tedy hlavně stav hrací plochy (kde jsou miny, s kolik minami políčka sousedí), vyhodnocení ukončení hry, počítání času a počítání zbývajících min
- **mainscreen.py** - hlavní obrazovka hry, řeší se zda game loop, ukončení hry, několik kolekcí sprites
- **sprites.py** - soubor, kde jsou třídy spritů + jeden SpriteSheet
- **startscreen.py** - úvodní obrazovka hry, kde je menu

Objektový model:

Řešení je rozděleno do několika tříd, z nichž ty zajímavější budou stručně popsány.

Game - třída přes kterou se hra spouští. Třída inicializuje pygame, řeší přechody mezi obrazovkami a spouští podkladovou hudbu v nekonečné smyčce.

StartScreen - třída pro obrazovku menu. Obsahuje 3 tlačítka pro vybrání obtížnosti, které mají hover effect a tlačítko pro ukončení hry. Po zvolení obtížnosti hráčem je tato obtížnost předána jako výstupní hodnota metody *show()*.

GameState - třída řeší stav hrací plochy. Má v sobě uložené dvě matice (List[List[]]), které popisují jednak podkladovou mapu (game_map) a jednak mapu, kterou vidí hráč (player_map). Podle těchto map se poté vykresluje hrací plocha. V game_map je zaznamenáno s kolika minami pole sousedí, případně, že je zda mina. V player_map je pro každé pole uložena buď konstanta pro “fog” (pole ještě nebylo odkryté) nebo “flag” (hráč si tam přidal vlaječku) nebo číslo, které opět říká, s kolika minami pole sousedí. Metoda reveal() je použita pro odkrytí vybraného pole. Pokud je v poli mina, hra končí a hráči se zobrazí game_map. Pokud v poli mina není, provede se překopírování hodnoty z game_map do player map (neboli místo “fog” vidí hráč po odkrytí pole číslo, s kolika minami dané pole sousedí). Dále třída obsahuje metody pro přidání vlaječky (*add_flag()*, pro zjistění uběhlého

času od začátku hry(*get_elapsed_time()*), pro zjištění počtu zbývajících min (*get_unrevealed_count()* podle vzorce počet min - počet vlaječek). Metoda *initialize_game_map()* se spouští po odkrytí prvního tlačitka a náhodně vygeneruje hrací plochu, tedy miny a podle toho spočítá čísla pro jednotlivá políčka (s kolika minami sousedí)

MainScreen - třída pro hlavní obrazovku. Na ní by se zvenčí měla volat pouze metoda *show()*, která zobrazí hlavní obrazovku a spustí hru. Třída si udržuje 3 kolekce spritů a sice jednotlivá pole, emotikon (ve skupině bude jeden prvek po celou dobu hry) a animace výbuchů (ve skupině bude nejvýše jeden prvek). Jako vstupní parametr dostává třída nastavení šířky, výšky (jako počet polí) a počet min. Na základě těchto parametrů se vygeneruje hrací plocha, rozměry jsou tedy dynamické. V game loopu se detekuje quit event a jestliže nastane, končí celá hra a zavírá se okno. Když se detekuje kliknutí levým tlačítkem myši, tak se pomocí detekce kolize najde sprite (pole), na které hráč kliknul a zavolá se metoda *reveal()* na instanci třídy GameState. Pokud byla pod polem mina, spouští se animace výbuchu a zvuk výbuchu. Kliknutí na pole s vlajkou nedělá nic. Pravé tlačítko myši přidává na pole vlajku (nebo odebírá pokud už tam byla). Na konci loopu se detekuje konec hry (ať už výhra nebo prohra) a případně se nastaví proměnná game_over a v dalším běhu game loopu se zobrazí game over obrazovka (pokud už doběhla animace výbuchu). Průběžně se také aktualizují hodnoty zbývajícího počtu min a uběhlého času. Po konci hry se také změní mapa z game state, která se hráči zobrazuje. Celou hru hráč vidí player_map, po konci hry je mu zobrazena celá podkladová mapa game_map (mapy mají stejný formát, takže se pouze vymění a metoda *update_bricks()*, která updatuje sprity kostiček hrací plochy, funguje pořád stejně)

Brick, **Emoji**, **Explosion**, **SpriteSheet** - třídy spritů, případně sprite sheet (ten používá třída Explosion pro nalezení správného snímku animace výbuchu)

SpecialSquareValues - třída obsahuje několik “magických” konstant, které se hodí k reprezentaci speciálních hodnot polí hrací plochy, tedy “fog”, “flag” a “mine”. Tím máme v game state v maticích, které reprezentují hrací plochu, pouze čísla, což je výhodné, protože čísly reprezentujeme to, s kolika minami dané pole sousedí.

Difficulty - třída pro reprezentaci obtížnosti hry. V řešení jsou tři potomci této třídy a sice **Beginner**, **Intermediate** a **Expert**, které nastavují parametry obtížnosti. Pokud bychom chtěli i “custom” obtížnost, tedy že hráč si zvolí šířku, výšku a počet min, tak můžeme tyto tři parametry poslat do konstruktoru této třídy a třídu poté předat do MainScreen, hrací plocha by se měla vykreslit podle těchto parametrů.