Programátorská dokumentace – Auralux

Nejdříve se spustí App.xaml + App.xaml.cs, ta připraví WPF a spustí **WindowMenu.xaml** a **WindowMenu.xaml.cs** – To je level menu.

Tam je případně nutné u background.ImageSource změnit cestu k obrázku mlhoviny dle místa souboru u vás (snad ne, měla by to být relativní cesta).

Kliknutím na dané tlačítko v souhvězdí spustí funkci Button_Click_X(), která zavolá MainWindow s parametrem X-1 což určuje jaký level se má tam načíst.

Class MainWindow

Jde již o samotnou hru – první se spustí MainWindow() ve třídě MainWindow – tam dojde k nastavení Timeru a jeho opakované volání funkce Engine, spustí se funkce Start().

Opět bude možná třeba nutné nastavit cestu k obrázku u background.ImageSource

Start()

V **Start()** dojde na základě toho jaký level byl passnutý při spouštění MainWindow k jeho načtení, tak že se spadne do daného ifu – tam dojde k nastavení proměnných a vybudovaní planet na plátně.

Po vytvoření planet dojde k přidělení jejich sousedů – to vytvoří vlastně graf, který je důležitý pro AI bota.

Pak již se spustí Engine a pak znova a znova….

Engine()

V Engine() se zavolá **PohybJednotek()** a **LoopPresPlanety()** a zvedne se counter počtu ticknutí (počet spuštění funkce engine) o +1. Po určitém počtu ticknutí v závislosti na rychlosti se spustí první if, který vytvoří jednotky. A po určitém počtu ticknutí se spustí druhý if, který řeší volání AI a případné určení *výhry nebo prohry.*

Dále třída obsahuje funkce, které zavolá plátno, když dojde k zmáčknutí klávesy či tlačítka myši:

Klik() - aktivováno levým tlačítkem myši

První se musí ověřit, zdalipak se kliklo na planetu (první if, uznává i klik na text a jeho hranici ne jen elipsu, protože to je taky na planetě (jedna se o její textový status o poctu jednotek).

Pak určí o jakou planetu šlo pomoci forcyklu a zjištění souřadnic planety a kliku → tři možnosti: kliklo se na vlastni planetu co nebyla vybrána → vybere se; kliknutí na planetu co je už vybrána → sníží vyber jednotek; kliknutí na planetu co je cizí a nějaká naše planeta je vybraná → odešle jednotky z vybrané planety.

Unklik() - aktivováno pravým tlačítkem myši

Pokusí se léčit planetu, určení jaké je stejné jako v Klik() zavolá se funkce Lecit() dané instance planety. Funkce je nevhodně pojmenovaná, protože jsem si nejdřív říkal ze to použiju na změnu vyberu, ale pak jsem se rozhodl pro léčeni a byl moc línej měnit název, protože člověk musí lozit i do toho xaml souboru.

Upgrade() - aktivováno scrollnutim kolečkem myši nahoru

Najde planetu stejně jako u léčit a zavolá to její funkci Upgrade().

PohybJendotek()

Zavolá u všech instanci jednotek PosunLetu() a pak to u hrace/bota vysype odpadky viz později.

LoopPresPlanety()

Zavola to u instanci planet funkci Infovypisjednotek() co vypíše na planete ten text s jejím statusem – jednotky, případně zdraví atd.

SpoctiOhrozeni()

Spočítá pro každou planetu, kolik nepřátelských jednotek okolo sebe (důležité pro AI bota)

Zamichej()

Zamíchá náhodně pole, používá se aby se bot rozhodoval pokazdy jinak (mícha se pole se sousedy planety u dané instance planety).

TestVyhry()

Viz nazev.

Class Player

Od této třídy dědí třída *Bot a Hrac*.

Jde o jednotlivé instance *hráčů* hry co drží planety. (slovem hráč rozumím buď vás jako člověka co to hraje za pc nebo bota, ne instanci třídy Hrac)

Důležitá je proměnná jednotky – tam jsou reference na všechny instance jednotek daného hráče.

TvorbaJednotek()

Loopne přes planety daného hráče a vytvoří jim to jednotky.

VysypatOdpadky()

Do *odpadky* se přidají instance jednotek co umřely – a pak se smažou při volání této funkce.

Class Hrac

Nepřidává nic navíc vůči player, volá stejný konstruktor.

Class Bot

AI()

První forsmycka zařídí to ze uvažujeme nyní planety, které nejsou daného bota, ale právě nepřátel, u nich se koukneme jestli mají okolo se pravě planety bota které mají dohromady dostatek jednotek, aby na ni zaútočili – pokud ano, tak s určitou pravděpodobností dojde k útoku.

Druhá forloopa loopne přes planety, které již ten bot vlastni, a koukne jestli daná planeta má dostatek jednotek, aby začala obsazovat prázdnou → pokud ano, s danou šanci to udělá.

Dale pak planeta posle své jednotky na planetu, která má vetší ohrožení vypočítaný.

Případně se podělí o jednotky jestli jich má vice s planetou co jich má méně a má stejné ohrožení.

Ty grafové algoritmy jsem neimplementoval s optimální časovou složitost, protože pocitam s tim, ze planet je malo, mene jak 15 a tedy kvadraticka slozitost je furt dobra, navíc stejne jedinou limitaci vykonu je Canvas a s tim ja nic udelat nemohu.

Celá AI je dost jednoducha a tohle byl prototyp, abych vedel jak bude co potreba, ale prekvapilo me ze to bohate staci na to, aby byla kompetetivni a byl jsem tedy pak moc linej ji vylepsovat. Jde teda porazit relativne snadou strategii, ze hrac nebude skoro vubec utocit a necha boty se porvat meti s sebou – to by slo snadno vyresit tak ze bot bude spis utocit na osobu co ma nejvice jednotek vubec, tohle je pripadne rozsireni.

Class Planeta

Její instance vzniknou ve Start() v MainWindow.

Má dost různých funkcí, ale ty jsou dost přímočaré a není úplně potřeba je tady vysvětlovat, protože to je pochopitelné z kódu a případných komentářů tam.

Důležité si jen uvědomit co je v ifech – zásadně se jedná o kontroly aby planeta němela víc jednotek/zdravi/levelu než je povoleno. (100*level je max jednotek pro tvoreni, zdravi je 100*level a level muze byt nejvyse maxlevel – s tim parametrem se vola jeji konstruktor.

Každá planeta má max level 2 v mých levelech, jde to změnit, ale pak by bylo dobre implementovat nějaky ukazatel toho jaky ma max level na upgrade, at se muze hrac rozhdonout jestli ji vubec chce obsadit.

Class Jednotka

Podobně jako v planetě by mělo být to pochopitelné z kódu – ty komplikovanejsi funkce zde vysvetlim.

Jakmile jednotka má umřít, je přidána do odpadků a až při sypaní je zabita – jak na platne tak jeji instance. Duvodem proc to nejde rovnou je ten, ze kdybych ji mazal rovnou tak modifikuju seznam pres který se loopuje a hodí to error, takže se sypou odpadky az po skonční te forsmycky.

PosunLetu()

Proměnné orb1 a orb2 jsou nasobitelne funkci sinu a cosinus ktere nam delaji kruhy pohyb a tedy jej meni na elipitkcy ve funkci NahodnyOrbit() se nahodne rozhodne jestli bude obihat zplostele nahore nebo dole – to je pro planety s level > 1.

Pokud jednotka neobiha a je na ceste na nejakou planetu tak se posouva jejim smerem a vola se funkce, ktera urcuje jestli uz dorazila → SrazkaSPlanetou()

SrazkaSPlanetou()

Hromada ifu, ktere zjistujou jestli se uz jednotka dostala na cilovou planetu a pak urci v jakym je stavu planeta (je cizi, moje, neobsazena atd.) a podle toho vola na planetě přislušné funkce.