Možnosti návrhového vzoru Entity-Component-System: případová studie

autor: Petr Kotáb | vedoucí: Mgr. Pavel Ježek, Ph.D. | 2024

Úvod

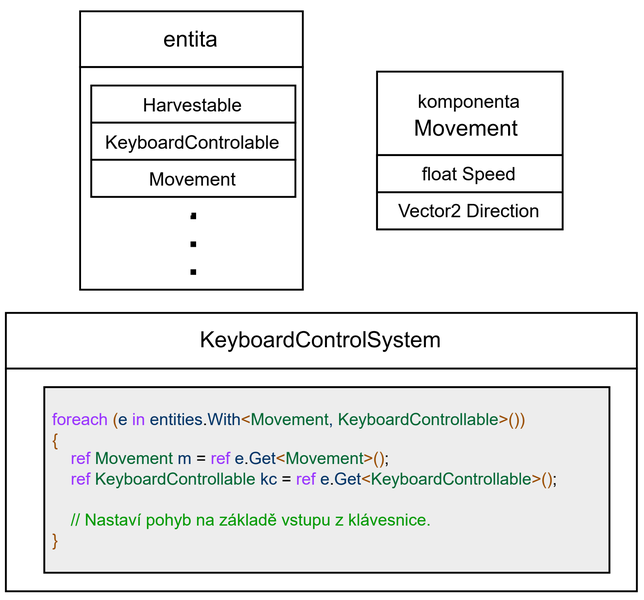
Entity-Component-System (ECS) je návrhový vzor pro tvorbu her. Jeho použití vyžaduje implementaci netriviální infrastruktury. Z toho důvodu spousta programátorů upřednostňuje použití ECS knihoven, které tuto infrastrukturu nabízejí.
Cílem této práce je usnadnit programátorům volbu vhodné ECS knihovny. Jedním z důležitých aspektů, podle kterého se chtějí vývojáři často orientovat, je srovnání výkonu mezi jednotlivými ECS knihovnami. Naše bakalářská práce se zaměří na tento aspekt.

Cíle práce

- Poskytnout srovnání výkonu ECS knihoven na netriviálním příkladě.
- Vytvořit ukázkovou hru nezávislou na konkrétní ECS knihovně.
- Provést měření populárních ECS knihoven pro C# za pomoci vytvořené hry.

Entity-Component-System

Hlavními stavebními kameny návrhového vzoru ECS jsou *entity*, *komponenty* a *systémy*.
Entita je objekt v herní scéně (například hráč, strom, nebo zvíře). Lze si ji představit jako kolekci *komponent*.
Komponenta popisuje schopnost nebo vlastnost. Například `Harvestable` *komponenta* přidává *entitě* možnost být sklizena jinou *entitou*, nebo `KeyboardControllable` přidává *entitě* možnost být řízena skrze klávesnici. *Komponenta* obsahuje pouze data a žádnou herní logiku.
Systém obsahuje herní logiku. Každý *systém* pracuje nad určitou množinou *komponent*. V každé své iteraci *systém* vezme všechny entity s určitou množinou komponent a provede nad nimi určité operace. Například `KeyboardControl` systém pracuje nad všemi entitami s komponentami `KeyboardControllable` a `Movement` a řídí logiku jejich ovládání.
Velkou výhodou ECS je vysoká flexibilita. Pokud bychom například chtěli aby, uživatel namísto hráčovy postavy ovládal zvíře, tak nám postačí z hráčovy postavy odebrat `KeyboardControllable` *komponentu* a přidat ji na zvíře.



Kontakt

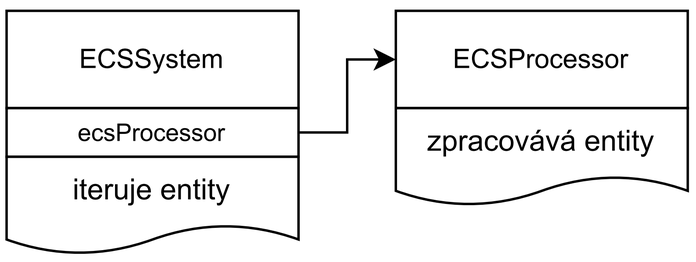
petrkotab99@gmail.com
github.com/pedryx/world-simulator

Hra

V práci jsme naimplementovali neinteraktivní hru, která simuluje vesnice a vesničany v otevřeném dvojrozměrném světě. Vesničané sbírají a zpracovávají suroviny, za které poté vesnice staví nové budovy. Svět je náhodně generovaný. Hráč v této hře zastává roli pozorovatele a na svět nahlíží kamerou shora.

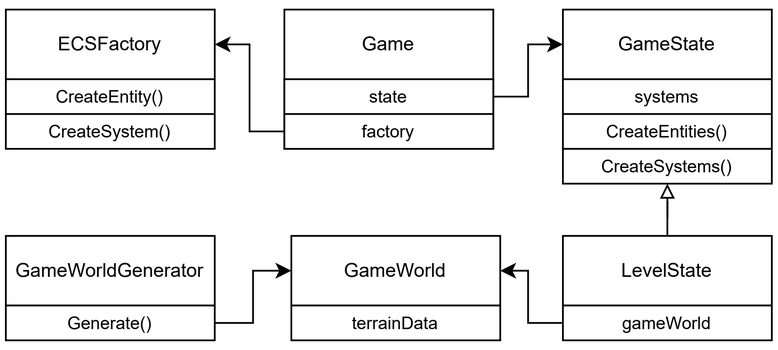
Abstrakční vrstva

Abychom mohli hru spouštět s různými ECS knihovnami, bylo nutné vytvořit abstrakční vrstvu, kterou hra využívá namísto konkrétní ECS knihovny.
Při tvorbě této vrstvy bylo nutné oddělit logiku iterace a zpracování entit, protože každá ECS knihovna iteruje entity jinak. Zavedli jsme třídu `ECSSystem`, kde každá dědící třída odpovídá za iteraci entit pro konkrétní ECS knihovnu. Tyto třídy obsahují instanci `ECSProcessor`, která je zodpovědná za zpracování entit. Např. můžeme mít `MovementProcessor`, který zpracovává pohyb a ten uvnitř `ArchSystem`, který implementuje iteraci entit pomocí ECS knihovny *Arch*.



Řešení

Pro tvorbu hry byla použita knihovna MonoGame. Hru reprezentuje instance třídy `Game` s dvěma hlavními členy: `state` a `factory`. `factory` je instance třídy, která dědí od `ECSFactory` a obsahuje metody pro vytváření tříd relevantních k ECS. Každá třída dědící od `ECSFactory` reprezentuje konkrétní ECS knihovnu. `state` je instance třídy dědící od `GameState`, která představuje herní obrazovku. Důležité členy této třídy zahrnují kolekci systémů a metody pro vytváření entit a systémů.
Hra má momentálně jednu herní obrazovku, kterou reprezentuje třída `LevelState`. Klíčovým členem této třídy je `gameWorld`, což je instance třídy `GameWorld`, která představuje herní svět. Tvorbu herního světa zajišťuje třída `GameWorldGenerator`.



Měřené knihovny

Používáme stejné knihovny jako v existujícím repozitáři *Ecs.CSharp.Benchmark*, který se zaměřuje pouze na jednoduché mikrobenchmarky, které nemusí poskytovat relevantní výsledky.
Do tohoto repozitáře jsou často přidávány další ECS knihovny. Nicméně, my jsme se soustředili na množinu knihoven, které repozitář obsahoval v době zahájení naší bakalářské práce.
Jednotlivé knihovny si rozdělíme do následujících kategorií:

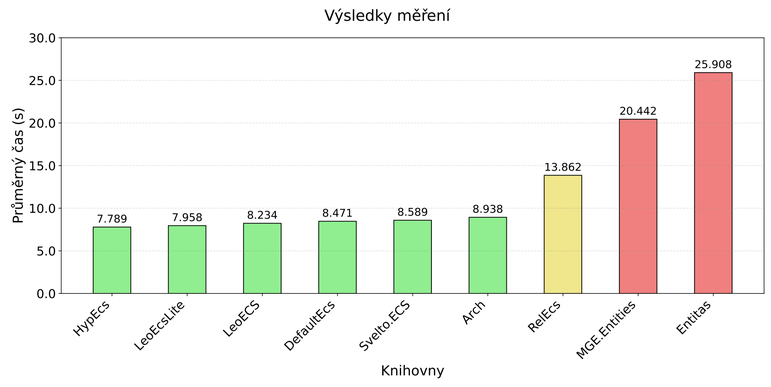
(1)	Knihovny, které dobře využívají cache.
(2)	Knihovny kombinující vlastnosti kategorií (1) a (3).
(3)	Knihovny, kde komponenty mohou být reprezentovány pouze jako třídy.

Stanovili jsme hypotézu, podle které knihovny z kategorie (1) dosáhnou nejlepšího výkonu díky efektivnímu využívání cache. Knihovny z kategorie (3) se naopak umístí nejhůře kvůli neefektivnímu využívání cache. Knihovny z kategorie (2) budou mít výkon mezi těmito dvěma kategoriemi.

Výsledky

Provedli jsme měření v různých scénářích a nyní si představíme výsledky pro jeden z nich:

knihovna	Průměrný čas	Chyba	StdDev
HypEcs	7.789 s	0.1542 s	0.3044 s
DefaultEcs	7.958 s	0.1585 s	0.3129 s
LeoECS	8.234 s	0.1647 s	0.3011 s
LeoEcsLite	8.471 s	0.1670 s	0.3177 s
Svelto.ECS	8.589 s	0.1711 s	0.3571 s
Arch	8.938 s	0.1775 s	0.4253 s
RelEcs	13.862 s	0.2707 s	0.3424 s
MonoGameExtended.Entities	20.442 s	0.2375 s	0.2222 s
Entitas	25.908 s	0.5035 s	0.6183 s



Výsledky jsou seřazené podle průměrného času. Jednotlivé knihovny jsou v grafu obarveny podle již zmiňovaných kategorií.
V různých scénářích nám vznikly menší rozdíly, ale relativní srovnání bylo vždy velice podobné. Ve všech scénářích se naše hypotéza potvrdila.
Je možné nahlédnout, že výsledky jednotlivých ECS knihoven se od sebe příliš neliší. Pokud by si tedy uživatel vybíral mezi ECS knihovnami a přišlo by mu, že pro něj je architektura nebo způsob využívání některé z knihoven kategorie (3) přínosnější nebo přehlednější oproti ostatním ECS knihovnám, stály by knihovny kategorie (3) také za zvážení.

Závěr

Podařilo se nám implementovat komplexnější hru na které jsme zhodnotili výkon jednotlivých ECS knihoven. Považujeme tedy všechny cíle za splněné.