JAKUB MELKA
ŠTĚPÁN POLJAK
MARTIN RŮŽIČKA
MARTIN URZA

DIVERGENT TERRITORY

- Online hra určená pro mnoho dlouhodobě hrajících hráčů (MMOG – massively multiplayer online game).

- Odehrává se ve sci-fi vesmíru, který obsahuje mnoho hvězd, slunečních soustav, planet i měsíců.
- Strategie s prvky RPG – navíc obsahuje z herního hlediska nepostradatelné simulace.

- Hráči ovládají základny a lodě, provádí výzkum, těží, bojují, obchodují, vyrábí, prozkoumávají vesmír.
- Cíl hry může být pro každého hráče jiný.

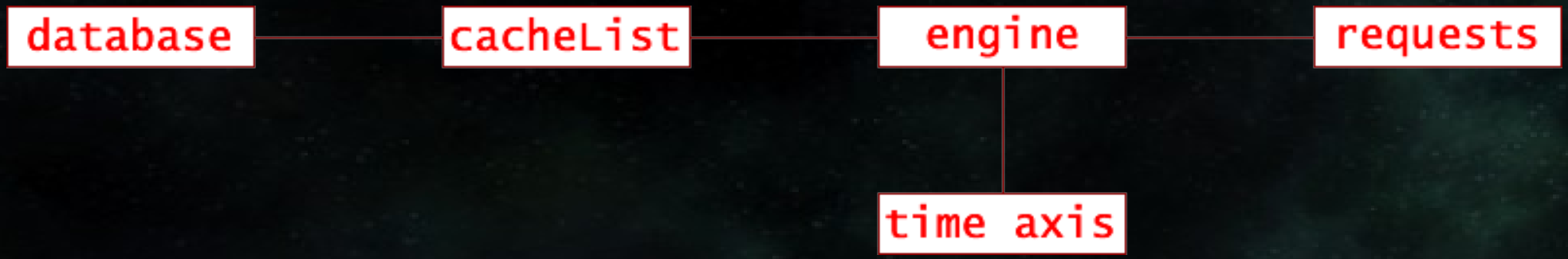
O MMOG OBECNĚ

- Určeny pro 64 a více hráčů - drtivá většina MMOG navržena pro tisíce až desetitisíce, nejúspěšnější hry hrají miliony.
- Dva zásadní rozdíly oproti klasickým krabicovým hrám:
  - Nemusí mít a většinou nemívají AI hráče – živí hráči soupeří se živými hráči.
  - Potřebují velmi výkonný server - musí postačovat desítkám tisíc registrovaných hráčů a stovkám až tisícovkám online hráčů.

VÝKON SERVERU

- Data nejsou pro každé použití vyžádána z databáze; server uchovává „vhodná“ data v paměti; přístup k nim zajišťují speciální struktury v konstantním čase.
- Většina výpočetně náročných operací probíhá paralelně.
  - Méně náročné operace jsou sériové kvůli ušetření implementačního času.
- Hra není řízena spojitou simulací; přepočítávání herních dějů jen při změně stavu - čas změny se spočítá předem.
- Kladli jsme důraz na efektivní kód - implementace v C++.

ARCHITEKTURA SERVERU



- Engine serveru řízen požadavky online klientů a událostmi na časové ose.
  - Požadavky hráčů jsou přijímány po síti.
  - Události na časovou osu si ukládá server sám; jedná se o kroky simulace s jednotlivými objekty.

- Data se v enginu nezískávají přímo SQL dotazy - databáze schovaná za cacheListy - data drží v sobě nebo je načtou z DB.
  - Pro server je to nerozlišitelné.

CACHE LISTY

- Třídy zajišťující přístup k datům - přijímají ID a vrátí objekt.
  - Objekty mohou být uloženy v paměti (listy na ně mají ukazatele), nebo v databázi.
    - Listy načtou data z databáze do paměti a nechají si ukazatel.
  - Zapouzdřují databázi a odstiňují engine od toho, kde a jak jsou data uložena.
    - Umožňují případně snadno přejít na jinou databázi.

- Implementace listů umožňuje přístup v konstantním čase, jsou-li data v paměti (jinak záleží na databázi).
- Každý typ dat má vlastní cacheList (základny, lodě, hráči, předměty, ….).

KOHERENCE CACHE LISTŮ

- Existuje mnoho způsobů nastavení pro čtení i zápis.
  - Čtení:
    - veškerá data jsou stále v paměti (a v databázi)
    - žádná data v paměti nejsou (vše je pouze v databázi)
    - data jsou v paměti od prvního použití (a zůstávají tam)
    - při nedostatku paměti jsou uvolněna nepoužívaná data
    - při nalogování hráče se načtou jeho objekty (lodě, ….)
  - Zápis:
    - jednou za čas se do databáze zapíší všechna data
    - jednou za čas se zapíší pouze změněná data
    - při každé změně jsou data ihned propsána do databáze
    - data se nezapisují (např. hvězdná mapa je konstantní)

ČASOVÁ OSA

- Mnoho dějů ve hře probíhá v čase - je třeba simulací.
  - Např.: pohyby lodí, těžba surovin, oprava poškozených předmětů, výroba předmětů, ….
  - Spojitá simulace by příliš zatěžovala server.

- Server pro každý děj spočítá čas příští změny.
- Následně do časové osy zanese událost „zkontroluj děj“.

- Události na ose vždy typu „zkontroluj, zda nemá být něco uděláno“.
  - Tedy nikdy ne typu „udělej něco“.

PROBLÉMY S UDÁLOSTMI

- V některých případech velmi obtížné spočítat čas následující změny.
  - Závisí-li na sobě cyklicky několik jevů, je těžké s nimi počítat a stanovit čas příští změny.
    - Př.: Androidi opravují rozvod energie, nicméně pro svou činnost sami energii potřebují (a čím opravenější rozvod je, tím více androidů jej může opravovat).
  - Pohyby lodí ve vesmíru jsou ovlivněny mnoha různými faktory (gravitačními poli, stavem motorů, ….). Kvůli složitosti fyziky jako takové je počítání časů změn dějů obtížné.
    - Například hledání „vhodného“ kořenu polynomu osmého stupně, vektorové a maticové výpočty a tak dále.

PARALELISMUS

- Architektura serveru umožňuje, aby jedna instance aplikace umožňovala běh více herních světů nad více různými databázemi.
  - V případě více procesorů a/nebo jader jsou tyto světy plně paralelní.

- V rámci jednoho světa je paralelismus jen částečný.
  - Výpočetně náročné operace probírají paralelně (tvorba zpráv, operace s textem, ….)
  - Aritmetické výpočty paralelizované nejsou.
    - Je to implementačně snazší.
    - Výkon to moc neovlivní (jedná se cca o 5% výpočetního času).

ARCHITEKTURA KLIENTA

- Klient slouží čistě k zobrazování toho, co mu server pošle, nemá sám ničemu rozumět, nic ověřovat, nic počítat.

- Klientskou část aplikace lze rozdělit na:
  - GUI – okna a dialogy, slouží k zobrazování a nastavování techničtějších dat hry, naprogramovaný v Qt.
  - Render – hvězdná mapa obsahující modely různých vesmírných objektů i hráčských lodí a základen.
    - **Tato část přesahuje rámec původní specifikace**, podle které mělo jít jen o symbolické zobrazení, ne OpenGL modely.
  - Komunikace – zvláštní vlákno určené pro komunikaci se serverem.

PROBLÉMY S HVĚZDNOU MAPOU

- Implementace hvězdné mapy klienta byla z několika důvodů velmi náročná.
  - Herní vesmír je obrovský; čtverec se stranami o délce cca 2000 světelných let (2^{54} km), pozice všech objektů jsou uloženy s přesností nejhůře na 0,5 km. Knihovní typy OpenGL mají přesnost mnohem menší - vyřešeno zavedením lokálního souřadného systému.
  - Velikosti a vzdálenosti v rámci herního vesmíru jsou realistické, což je pro zobrazování velmi nevhodné.
    - Řešení problému:
      - Obrovský (exponenciální) zoom.
      - Možnost zvýrazňovat vesmírné objekty a lodě bez ohledu na měřítko (zvětšením na nějakou konstantní velikost).

PROBLÉMY S MODELY

- Modely, které jsou volně ke stažení, typicky nemají „osekané“ verze s menším počtem polygonů.
  - Což neumožňuje škálovatelnost „grafických detailů“.
  - Vyřešeno používáním modelů, které rovnou obsahují relativně málo polygonů.
- Většina z modelů hvězdných lodí, které lze bezplatně legálně použít, není otexturována.
  - Výsledkem je, že máme celkem málo (šest) tříd lodí.
  - Pro přidání dalších modelů není ani třeba modifikovat kód serveru (stačí pozměnit číselníkové tabulky v db).
- Definitivně by problém řešil profesionální modelář

PARAMETRIZACE HRY

- Na rozdíl od „krabicových“ her, které si hráči koupí, zahrají a odloží, jsou online hry úspěšné a dobré jen tehdy, když u nich hráči zůstávají.
  - Žádná hráčská základna nezůstane moc dlouho u hry, která se nevyvíjí (mnohokrát empiricky ověřeno).
- Online hry je nutno tvořit snadno modifikovatelné.
  - Hru samotnou lze do značné míry parametrizovat jen změnami číselníkových tabulek databáze.
  - Ještě větších změn lze dosáhnout modifikací konstant.
  - Objektový model je lehce rozšiřitelný o nové předměty.
  - Engine je navržen dokonce tak, že by jej šlo relativně snadno přepsat na úplně jinou hru podobného typu.

LADÍCÍ PROSTŘEDKY

- Koncepce hry předpokládá dlouhodobé hraní (plný rozvoj hráče trvá dlouhé měsíce, možná krátké roky).
  - Tolik času pochopitelně při testování není.

- Herní čas běží defaultně 30x rychleji než reálný, je však možné ho ještě urychlit (změnou konstanty).
  - Přílišné urychlení ale neřeší vše - čas ve hře sice uplyne, ale hráč nestihne vykonat tolik akcí, kolik se předpokládá.

- Hra umožňuje založení „nových“ hráčů, kteří už mají rozvinutou vědu, bohaté vybavení, mnoho lodí i základen.
  - To lze dělat pohodlně přes administrátorské rozhraní.

TECHNICKÉ DETAILY

Programovací jazyk serveru	C++
Programovací jazyk klienta	C++
Programovací jazyk administrátorského rozhraní	C#
Řádek kódu	127 181
Vývojové prostředí	Visual Studio 2010
Platforma	MS Windows
Databáze	Firebird
Síťová komunikace	TCP sockety

DĚKUJEME ZA POZORNOST

Rádi zodpovíme vaše dotazy

?