Java is a new C

Мы далеки как «да» и «нет»?

Считается, что Си и Ява – враги?

- В мире разработчиков на Си Ява считается наиболее мерзким, богопротивным и кардинально далёким от Си языком, на котором пишут люди с пёсьими головами. Программы на Яве работают чуть медленнее, чем программы на Бейсике для ПЭВМ Агат и отвратны по сути своей. Ересь!
- В мире Явы Си тоже не очень любят. За segmentation faults.

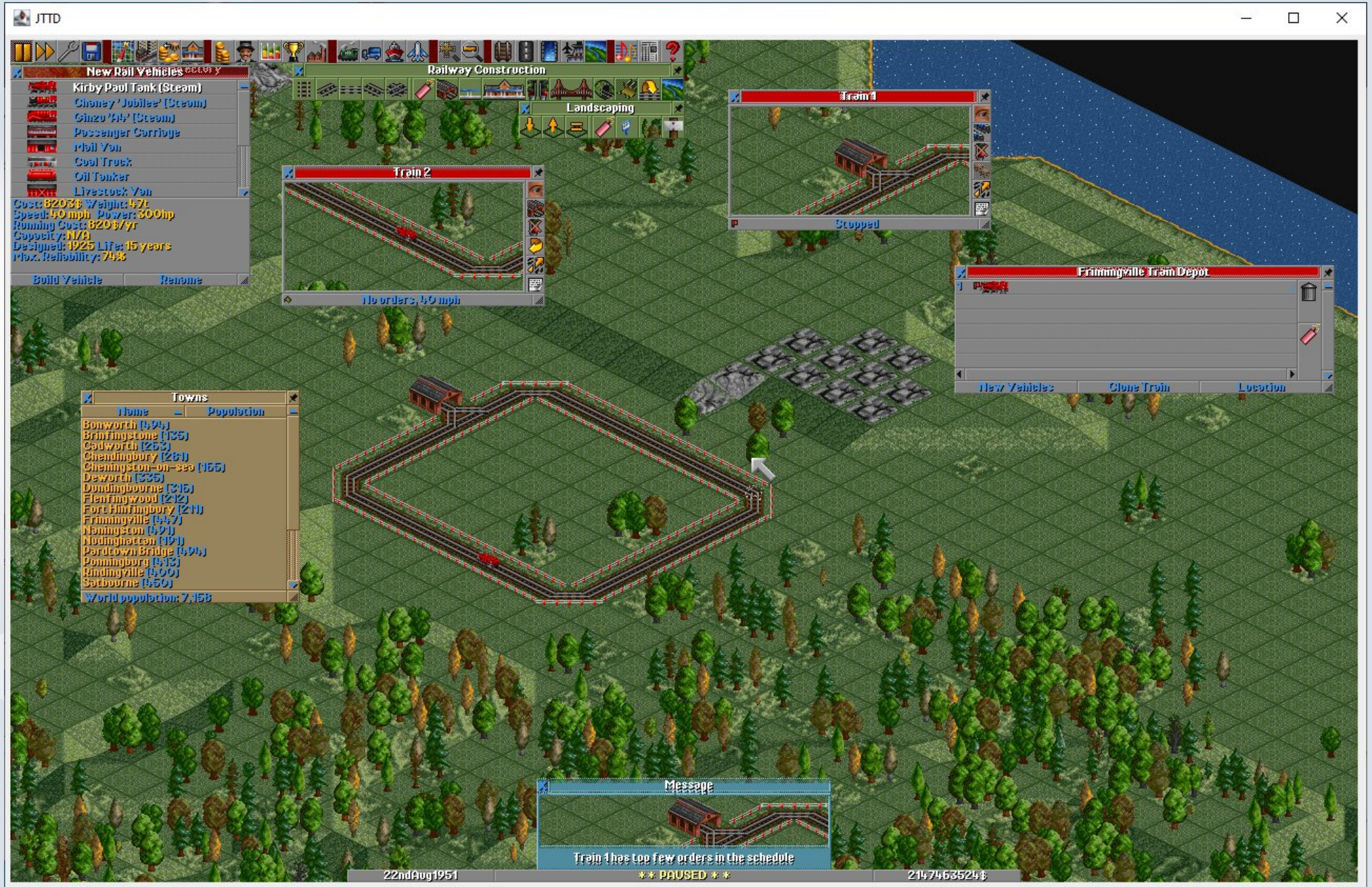
Ява – ближайший потомок Си

- Де факто Яву можно рассматривать как потомка языка Си, причём настолько близкого, что возможен перенос кода из Си в Яву методом коррекции по месту.

- Я приведу практический пример, в котором исходный текст игры на Си размером в 300 тысяч строк был конвертирован в чистую Яву с затратами менее чем 1 ч/м.

Исходная точка

- Transport Tycoon Deluxe – написана на ассемблере в 90-х.
- В 2004 дизассемблирована и конвертирована в Си. С полным сохранением структур данных и, насколько я могу судить, алгоритмов.
- Итого - написана в 16-битной ассемблерной манере, с жуткими битовыми полями, в которых трактовка одних бит зависит от значения других, с «ручной» 24-битной fixed point арифметикой, танцами с пойнтерами и бубном (обращение к «обрамляющей» структуре через отрицательное смещение, всё как мы любим) и всеми иными тяжкими, на которые идут ассемблерные программисты, чтобы сэкономить ещё пару байт памяти.
- Что, наверное, было уместно в 1994, но сейчас от всего этого – волосы дыбом.


JTTD
New Rail Vehicles
Kirby Paul Tank (Steam)
Cheney 'Jubilee' (Steam)
Ginzu '44' (Steam)
Passenger Carriage
Mail Van
Coal Truck
Oil Tanker
Livestock Van
Cost: 8203$ Weight: 47t
Speed: 40 mph Power: 300hp
Running Cost: 820$/yr
Capacity: N/A
Designed: 1925 Life: 15 years
Max. Reliability: 74%
Build Vehicle
Rename
Railway Construction
Landscaping
Train 2
No orders, 40 mph
Train 4
Stopped
Framingville Tram Depot
New Vehicles
Clone Train
Location
Towns
Name
Population
Bonworth (494)
Brinfingstone (135)
Cadworth (263)
Chendingbury (281)
Chenington-on-sea (165)
Deworth (335)
Dondingbourne (316)
Flenfingwood (212)
Fort Hinfingbury (211)
Frinmingville (447)
Nanningston (491)
Nudinghattan (191)
Pardtown Bridge (494)
Ponningbury (413)
Rindingville (400)
Satbourne (450)
World population: 7,158
Message
Train 1 has too few orders in the schedule
22ndAug1951
** PAUSED **
2147463524$

Цель

- Доказать, что Ява очень близка к Си и код на Си можно повторить на Яве 1:1
- Получить pure java реализацию
- По возможности упростить код там, где это бесплатно: управление памятью, код, необходимый для поддержки разных ОС, контейнеры и пр.

Чего я не хотел

В силу ограниченности времени:

- Оптимизаций. Напротив, в некоторых местах оптимизация была безжалостно снесена, чтобы снизить риски ошибок.
- Рефакторинга. Просто потому что по хорошему там надо рефакторить 100% кода и структур данных, см выше про ассемблерный подход.

Общая схема работы

- Код тупо пофайлово перебрасывался из .c в .java
- Прогонялась пачка скриптов для пакетного редактора sed, которая делала стандартные автозамены. Например, удаляла все символы '*', заменяла const на final и т.д.. Скрипты пополнялись и, наверное, к 10-му файлу выполняли ощутимую часть работы. Правда, потом кое-где приходилось делать обратные замены, к примеру 'finalruct' на 'construct'. :)
- Для глобальных функций, переменных и констант скрипты добавляли префикс объекта или класса, в который эти переменные перекочевали.
- Наконец, оставшийся в «красной зоне» код правился вручную по месту.

Не проблема: Указатели

Элементарный класс решил все проблемы:

```
public class ArrayPtr<ItemType> implements IArrayPtr
{
        private final ItemType [] mem; // real mem
        private int displ;                  // current displacement
...
}
```

В частности, вся работа с графикой работает через ByteArrayPtr – несмотря на страшный вид кода, оптимизатор отработал на пять и никакой деградации не просматривается.

Не проблема: Возврат значений через указатель

```
modify( int[] x )
{
  x[0]++;
}
```

Нежданная боль – unsigned

Изначально игра была написана на ассемблере, и беззнаковый байт использовался в коде очень часто. При первичном портировании он был заменён на обычный инт, который полностью перекрывает нужный диапазон значений, но оказалось, что есть места, где переполнение 8 или 16-битного беззнакового используются в логике кода. Пришлось выявлять такие места и добавлять &= 0xFF.

Видите переполнение? А оно есть.

```
static bool ShipAccelerate(Vehicle *v)
{
	uint spd;
	byte t;

	spd = min(v->cur_speed + 1, v->max_speed);

	//updates statusbar only if speed have changed to save CPU time
	if (spd != v->cur_speed) {
		v->cur_speed = spd;
		if (_patches.vehicle_speed)
			InvalidateWindowWidget(WC_VEHICLE_VIEW, v->index, STATUS_BAR);
	}

	// Decrease somewhat when turning
	if (!(v->direction & 1)) spd = spd * 3 / 4;

	if (spd == 0) return false;
	if ((byte)++spd == 0) return true;

	v->progress = (t = v->progress) - (byte)spd;

	return (t < v->progress);
}
```

Оператор “запятая”

Красивый код:

```
( bonus += 10, age > 10 ) ||
( bonus += 20, age > 5 ) ||
( bonus += 40, age > 2 ) ||
( bonus += 100, true );
```

Оператор “запятая”

Некрасивый код:

```
//if (!tile.IsTileType(TileTypes.MP_RAILWAY) || ((dir = 0, tile.getMap().m5 != 1) && (dir = 1,
_tile.getMap().m5 != 2)))
//          return Cmd.return_cmd_error(Str.STR_1005_NO_SUITABLE_RAILROAD_TRACK);

if (!tile.IsTileType(TileTypes.MP_RAILWAY))
          return Cmd.return_cmd_error(Str.STR_1005_NO_SUITABLE_RAILROAD_TRACK);

if(tile.getMap().m5 == 1)
          dir = 0;
else if(_tile.getMap().m5 == 2)
          dir = 1;
else
          return Cmd.return_cmd_error(Str.STR_1005_NO_SUITABLE_RAILROAD_TRACK);
```

Нежданная боль - Enum

В языке Си теги enum – целые константы. И их можно проверять битовыми операциями. В Яве – объекты. И их – нельзя. Самым простым решением было заменить enum на просто целые константы, что соответствует духу Си и не требует переписывания кода.

Макросы

Макросов в Яве нет, как нет и неявного взятия адреса аргумента при вызове функции, поэтому макросы, которые модифицируют свой аргумент были самой большой болью

Было:
SET_BITS(tile.getMap().m2, 0, 4, new_ground);

Стало:
tile.getMap().m2 = BitOps.RETSB(tile.getMap().m2, 0, 4, new_ground);

Конечно, этот код – тяжёлое наследие оригинального ассемблерного кода, который хранил свойства объектов в битовых полях внутри байта, и в реальном современном коде такое вряд ли встретится.

Целое как булево значение

Это решается заменой типа, если целое используется только как булево значение. Но масса проверок именно целочисленного значения на ноль / не ноль потребовала массовой же замены “if(“ на “if(0 != (“

Память

- Версия на си требует 8.5 мегабайт памяти.
- Ява – 178. Поначалу расстроило, верно? :)
- По факту из них 137 – это новые звуки в высоком разрешении, которые приложение загружает при запуске.
- Вторая часть избыточного футпринта – такая же массовая загрузка графики при запуске – я сознательно выкинул все механизмы частичной загрузки, и, конечно, это сказалось.
- Объём избыточных затрат на графику, скорее всего – порядка 12 мегабайт.
- Итого затраты памяти – 29 мегабайт, что, конечно, всё равно втрое больше оригинального, но разумно предположить, что существенная часть – накладные расходы рантайма и они имеют аддитивный, а не мультипликативный характер.

Проверить это можно бы методом портирования системы с на порядок большими требованиями к памяти, но такой длинный отпуск я себе обеспечить не смогу. :)

Скорость работы

- Серьёзных бенчмарков не делалось. Визуально даже на самых тяжёлых режимах (быстрый и длительный скроллинг полной карты мышкой на весь экран) производительность не вызывает никаких вопросов.
- При этом стоит учесть, что версия на Си сильно проседает по производительности в некоторых режимах игры (очень большое количество кораблей в игре) в силу сложности алгоритмов нахождения пути в открытом (море) пространстве.

То есть, в целом, производительность в данном случае – скорее вопрос качества алгоритма, а не языка реализации.

Итоги

- Язык программирования Ява с алгоритмической точки зрения является достаточно прямым наследником языка Си и прямое портирование кода на Си в Яву более чем возможно.
- Ожидаемые проблемы, связанные с наличием в Си адресной арифметики в реальном коде не проявляются.
- Невозможно портировать код, в котором указатели меняют тип и пересекают границы объектов, но, как правило, существуют нативные инструменты реализации той же функциональности, которые на порядок проще оригинальных реализаций на Си. Примеры: аллокация и деаллокация сложных структур (деаллокация в Яве не требуется), сериализация объектов (есть встроенные механизмы рантайма Ява).

Трудоёмкость портирования по факту оказалась весьма умеренной – порядка 10 дней на 100 000 строк.

Что не вошло

Глубокое взаимодействие с целевой ОС, например, работа с системными вызовами windows/linux. Игра требовала всего 5 интерфейсов – фрейм буфер, клавиатура, мышь, midi и wav аудио. Все они присутствуют в Яве из коробки и очень легко интегрируются.

Выводы по методике

- Ни в коем случае не рефакторить по ходу. Код должен оставаться как можно более близким к оригинальному. Работа делится на две фазы – портирование (код не компилируется) и исправление ошибок (код компилируется, но не работает). Если код в процессе меняется, вторая фаза превращается в ад. Должно быть возможно посмортеть на оригинал на Си и сравнить логику. Исключение – контейнеры можно внедрять сразу, если логика оригинальных контейнеров очевидна и банальна.

- Юнит тесты для оригинального кода! В моём случае их не было, и это проблема.

- Если прямой перенос подсистемы невозможен и явно потребуется переписывание кода, её лучше выключить (если такое возможно) и отложить на потом. В моём случае это были save/load, музыка и звуки.

Верификация

Перед этим выступлением я провёл повторный эксперимент. Портирование на Яву редактора FTE. Порядка 70000 строк. Исходный код – C++.

Итого – неделя до ожившего кода, две до работоспособного редактора. Вечерами!

Главная проблема с C++ – деструкторы, заменяются на Closeable.

Почему я этим занялся

Это моя общественно полезная работа. Субботник.

В России очень много людей из прошлого века, для которых Си – альфа и омега. (Я сам такой был)

Эти люди сегодня принимают решения и выбирают направления развития. И часто выбирают Си как язык разработки. Это ощутимо вредит экономике страны. Я ставил перед собой задачу показать, что Си и Ява – не два противоположных полюса мира программирования, а очень близкие языки.

Контакты

Проект: https://github.com/dzavalishin/jdrive/

Дмитрий Завалишин

http://dzsystems.com/

Digital Zone, E-Legion, Aprentis, RnDFlow