[RElog] – Oel (Pompowacze) port z C64 (część 3) (sheadovas/artykuly/relog/oel-port/relog-oel-pompowacze-port-z-c64-czesc-3/)

Maj 06, 2017 / Oel Port (sheadovas/category/artykuly/relog/oel-port/)

Wstępny kod portu

Hej, od ostatniego wpisu o porcie do gry *Oel* minęła już chwila, mimo niezbyt dużej ilości wolnego czasu na ten projekt znalazłem chwilę aby coś przy nim podłubać, a dzisiaj chciałbym przedstawić co udało mi się osiągnąć Mówimy o zmianach do commit'a [0c41b6f (https://github.com/sheadovas/oel-port/commit/0c41b6f8e7ac71350673ad1a4f8fcd4d4596ee3b)].

Przede wszystkim postanowiłem zabrać się za mapę zmiennych, jak pamiętacie z poprzedniego wpisu, to byłą dość słabo uzupełniona. Aby ją uzupełnić można było wybrać dwa podejścia:

1. Długa i nieciekawa analiza, także podczas runtime
2. Zgadywanie przez tłumaczenie stringów.

Oczywiście wybrałem prostszą opcję, a więc (2).

Tłumaczenie stringów

Oryginalny kod źródłowy został napisany przy użyciu mieszanego języka niemieckiego z odrobiną angielskiego. Szczególnie w niemieckim zdarzają się błędy w słowach i drobne przekręcenia, co nie pomaga w tłumaczeniu przy praktycznie zerowej znajomości tego języka.

Jak być może pamiętacie, to już wcześniej zrobiłem sobie pliczek [notes/strings.txt (https://github.com/sheadovas/oel-port/blob/0c41b6f8e7ac71350673ad1a4f8fcd4d4596ee3b/notes/strings.txt)] do którego zrzuciłem wszystkie stringi z kodu źródłowego, zacząłem w nim także zawierać „ręcznie" robione tłumaczenia tekstów.

Przy użyciu jakże wyrafinowanej metody: „skopiuj tekst do schowka, wklej do Google Translate, skopiuj tłumaczenie i wklej do pliku" dociągnąłem do około ~100 tekstów (na ~250). Możecie sobie tylko wyobrazić jak taka robota jest nudna i czasochłonna, dlatego znudzony tą robotą napisałem sobie prosty skrypt w Pythonie który zajął się automatycznym tłumaczeniem tekstów:

translate.py — Python

```python
import goslate

raw = [
" gehoert ihnen nicht!",
" geht nicht!",
" gelungen ",
----[ CUT SNIPPET ]----
"zum verkauf ansteht.",
" zur zeit keine pumpenherstellungsfirma",
" zuviel"
]


gs = goslate.Goslate()
for s in raw:
    t =  gs.translate(s, 'en', 'de')
    print "%s -> %s" % (s, t)
```

Dla większej czytelności wyciąłem część stringów do przetłumaczenia. Sam skrypt (umieszczony pod ścieżką [scripts/translate.py (https://github.com/sheadovas/oel-port/blob/0c41b6f8e7ac71350673ad1a4f8fcd4d4596ee3b/scripts/translate.py)]), wykorzystuje jak się później okazało niespieraną już bibliotekę *GoSlate*, na szczęście mi zadziałała poprawnie.

Całość wymagała jeszcze paru ręcznych poprawek (z racji pojawienia się literówek w oryginalnych tekstach), samo wykonanie skryptu z ręcznymi poprawkami skróciło czas tłumaczeń z wielu godzin do ~10min łącznie z napisaniem skryptu (*Praise the Python!*).

„Finalna” lista stringów wygląda następująco:

notes/strings.txt

```
" 1. oelfelder erwerben."                          -> "1. Purchase oilfields"
" 1 = pumpenpreis"                                 -> " 1 = pump price"
" 20000 Dollar. Du musst jetzt jedes Jahr"         -> " $20000. You have to go (FM:earn) now every year"
" 2. pumpenhersteller werden."                     -> " 2. Manufacturers"
" 2 = tankwagenpreis"                              -> " 2 = Tank price"
" 3 = bohrgestaengepreis"                          -> " 3 = Drill price"
" 3. tankwagenhersteller werden."                  -> "3. tank truck manufacturers."
" 4. bohrgesellschafter werden."                   -> "4. (FM) Become a drilling company leader"
" 5000 Dollar abbezahlen."                         -> "Payoff: $5000"
" 5. sabotage betreiben."                          -> "5. Sabotage operation"
" 6. und natuerlich alles zusammen."               -> "6. And of course all together."
" 7. oder gar nichts von allem."                   -> "7. Or nothing at all."
"a = bohrgesellschaft"                             -> "a = Drilling company"
" agenten anwerben und fremdes oel-"               -> " agents attract and foreign oil-"
" alles klar!"                                     -> " all clear!"
" ausser kraft setzen. = f7"                       -> " (FM) override. = F7"
"bay. pumpe eg "                                   -> "bay. pump eg "
"benzinacker "                                     -> "Petrol station"
" b e s i t z e r "                                -> " o w n e r "
"besitzer:"                                        -> "owner: "
" betrieb setzen. = f3"                            -> "owner seat. = F3"
" betr. pumpenfabrikanfrage"                       -> "(FM) Pump factory request"
" bisherige bohrtiefe :"ab(i)"m"                   -> " previous drilling depth %dm"
"bitte druecken sie eine taste"                    -> "Please press a button"
"black gold "                                      -> "Black gold"
" bohrfirma durch oelueberschwemmung"              -> " (FM)Drilling company by oil flooding"
" bohrgesellschaftskaufangebote "                  -> " Purchase Drilling Company"
" bohrgestaenge noch fuer:"                        -> " Drilling still for:"
"bohr & pump & sohn"                               -> "Drill & Pump & Son"
"bohrung auf feld"                                 -> "Drill on field"
" bohrung unmoeglich, da kein gestaenge"           -> "Bore impossible, because no one left on stuck"
" b = pumpengesellschaft"                          -> " b = Pump Company"
" CK"                                              -> " (FM) CK"
```

Na koniec tego rozdziału mam ogromną prośbę do osób znających język niemiecki (jeżeli znacie takie osoby to możecie przekazać moją prośbę do nich): powyższy listing tłumaczeń na pewno w wielu miejscach jest mało doskonały i chciałbym Was prosić o poprawienie powyższych tłumaczeń (tłumaczenia na język polski też są mile widziane).

Najbardziej aktualne tłumaczenia tekstów znajdują się pod [tym linkiem (https://github.com/sheadovas/oel-port/blob/master/notes/strings.txt)], jeżeli ktoś jest zainteresowany tłumaczeniem lub zna osoby gotowe podjąć się tłumaczenia to prosiłbym o kontakt (mail, fb, komentarz).

Mapa zmiennych

Gdy już mamy tłumaczenia tekstów, to samo zgadywanie funkcjonalności zmiennych stało się znacznie prostsze. Skupiłem się głównie na tablicach, które mogą przechowywać dość ważne informacje opisujące graczy, rafinerie, itp.

Samo zgadywanie odbywało się dość schematycznie:

1. Wybierz zmienną, np *ta*
2. Wyszukaj jej powtórzenia w kodzie przy użyciu komendy: `grep -C 3 listing.bas -e "ta("`
3. Znajdź string opisujący zmienną w okolicach wystąpień znalezionego miejsca użycia zmiennej

variables.txt

```
ARRAYS:
#DEFINE OIL_FIELDS_NUM 13
ab(OIL_FIELDS_NUM)          -> int[OIL_FIELDS_NUM] drilling_depth
bd(OIL_FIELDS_NUM)          -> int[OIL_FIELDS_NUM] drilling_price_for_500m
bm(OIL_FIELDS_NUM)          -> int[OIL_FIELDS_NUM] drilling_meters

bf$(OIL_FIELDS_NUM)         -> string[OIL_FIELDS_NUM] drilling_scrn_strings

bp(OIL_FIELDS_NUM)          -> int[OIL_FIELDS_NUM] drill_price
bt(OIL_FIELDS_NUM)          -> int[OIL_FIELDS_NUM] maybe_drill_meters_limit
fb(OIL_FIELDS_NUM)
f(OIL_FIELDS_NUM)
fm(OIL_FIELDS_NUM)          -> int[OIL_FIRLDS_NUM] fuel_source_limit
pa(OIL_FIELDS_NUM)          -> int[OIL_FIELDS_NUM] amount_of_pumps
ta(OIL_FIELDS_NUM)          -> int[OIL_FIELDS_NUM] amount_of_tanks
rp(37)                      -> int[37] refinery_price
gf(12)                      -> int[12] shipping_amount
gg(12)
p(?)                        -> int[?] pump_price

pf$(?=2)                -> string[2] corpo_name
pf(?=2)                 -> int[2] corpo_price

tg$(?=4)                -> string[4] firm_name
tp(?=4)                 -> int[4] firm_price
tw(?=4)                 -> int[4] truck_number
b$(?=3)                 -> str company_name
bg(?=3)                 -> int maybe_company_cost
ks(sz)                  -> int[players_count] player_money
sn$(sz)                 -> str[players_count] player_name
bk(sz)                  -> int[players_count] player_bankloan
```

Tym sposobem załatwiłem większość tablic, chciałbym też zauważyć, że część zmiennych wciąż nie jest zawarta w pliku [notes/variables.txt (https://github.com/sheadovas/oel-port/blob/0c41b6f8e7ac71350673ad1a4f8fcd4d4596ee3b/notes/variables.txt)], ale czas na nie przyjdzie z czasem w miarę potrzeb.

Piszemy port

Wreszcie dochodzimy do miejsca w którym chciałbym się z Wami podzielić bardzo wczesnym kodem portu do gry. Jak (chyba) wspominałem w pierwszym poście będę kod pisał w C++ przy użyciu SFML (biblioteka graficzna), jako IDE wybrałem ostatecznie [CLion (https://www.jetbrains.com/clion/)]. Postanowiłem też lekko zmodyfikowaną wariację [stylu pisania Google (https://google.github.io/styleguide/cppguide.html)].

Kod portu znajduje się w repo pod ścieżką [src/ (https://github.com/sheadovas/oel-port/tree/0c41b6f8e7ac71350673ad1a4f8fcd4d4596ee3b/src)]. Tutaj będę przedstawiał i omawiał jedynie fragmenty kodu, ciekawscy pełnego kodu muszą kierować się na repo.

Myślę, że jako tako udało mi się odwzorować klasę Gracza, który prawdopodobnie wzbogaci się jedynie o metody do operowania na pieniądzach, posiadanym majątku, etc.

Player.h C++

```cpp
class Player {
public:
    std::string name;
    int money;
    int bank_loan;

    Player(std::string name, int starting_money);
};
```

Powyższa klasa operuje oczywiście na zmiennych znalezionych w oryginalnym kodzie gry, dla przypomnienia:

```
ks(sz)                -> int[players_count] player_money
sn$(sz)                -> str[players_count] player_name
bk(sz)                 -> int[players_count] player_bankloan
```

Jak widzimy zaproponowane nazwy zmiennych w pliku *variables.txt* nie do końca pokrywają się z tymi w kodzie, powód jest prosty: w kodzie piszemy obiektowo, wg innych zasad więc nazwy są inne. Te w pliku *variables.txt* mają jedynie za zadanie ułatwić analizę kodu w języku BASIC oraz ewentualne cofnięcie się do pierwotnych nazw.

Oprócz tego zacząłem pisać sobie kod emulujący warstwę graficzną, w tym celu przepisałem wszystkie kolory z Commodore na ich odpowiedniki w SFML (do tego posłużyłem się [tabelą kolorów (https://www.c64-wiki.com/wiki/Color)]):

colors.h C++

```cpp
#include <SFML/Graphics/Color.hpp>

namespace c64 {
    static const sf::Color BLACK = sf::Color(0, 0, 0);
    static const sf::Color WHITE = sf::Color(255, 255, 255);
    static const sf::Color RED  = sf::Color(136, 0, 0);
    static const sf::Color CYAN = sf::Color(170, 255, 238);
    static const sf::Color VIOLET = sf::Color(204, 68, 204);
    static const sf::Color GREEN = sf::Color(0, 204, 85);
    static const sf::Color BLUE = sf::Color(0, 0, 170);
    static const sf::Color YELLOW = sf::Color(238, 238, 119);
    static const sf::Color ORANGE = sf::Color(221, 136, 85);
    static const sf::Color BROWN = sf::Color(102, 68, 0);
    static const sf::Color LRED = sf::Color(255, 119, 119);
    static const sf::Color GREY1 = sf::Color(51, 51, 51);
    static const sf::Color GREY2 = sf::Color(119, 119, 119);
    static const sf::Color LGREEN = sf::Color(170, 255, 102);
    static const sf::Color LBLUE = sf::Color(0, 136, 255);
    static const sf::Color GREY3 = sf::Color(187, 187, 187);
};
```

Z ciekawszych rzeczy to rozpisałem sobie także szkielet pod inicjalizację zmiennych oraz pętle wewnątrz gry (menu, główna pętla).

Game.cpp C++

```cpp
int Game::Run() {
    /* Init section (original) */
    if(init_values_()) {
        return 1;
    }

    rectangle_screen_.setOutlineColor(c64::BLACK);
    rectangle_screen_.setFillColor(c64::RED);
    color_clear_ = DEFAULT_CLEAR_COLOR;

    if (prepare_consts_to_load_()) {
        return 2;
    }

    if(load_graphics_module_(640, 480) /* FIXME use other values */) {
        return 3;
    }

    if(show_menu_()) {
        return 4;
    }

    /* Game loop */
    // TODO game loop

    return 0;
}
```

Jeżeli zwrócicie uwagę, to podczas grania na ekranie główny ekran gry otoczony jest [ramką (https://pl.wikipedia.org/wiki/Commodore_64#/media/File:C64_startup_animiert.gif)], samą ramkę i tło głównego ekranu postanowiłem emulować zwykłym prostokątem z obwódką:

Game.cpp C++

```cpp
int Game::load_graphics_module_(unsigned int scrn_width, unsigned int scrn_height) {
    if(!font_commodore_.loadFromFile("../resources/PetMe.ttf")) {
        return 1;
    }

    scrn_min_  = sf::Vector2f(0.0f, 0.0f);
    scrn_max_ = sf::Vector2f(float(scrn_width), float(scrn_height));
    scrn_center_ = sf::Vector2f((scrn_max_.x - scrn_min_.x) / 2.0f, (scrn_max_.y - scrn_min_.y) / 2.0f);

    rectangle_screen_.setSize(sf::Vector2f(scrn_width, scrn_height));
    rectangle_screen_.setOrigin(scrn_center_);
    rectangle_screen_.setPosition(scrn_center_);
    rectangle_screen_.setOutlineThickness(-64.0f); // FIXME find proper value

    window_.create(
            sf::VideoMode(scrn_width, scrn_height),
            "Oel - Port by shd",
            sf::Style::Close | sf::Style::Titlebar
    );
    return 0;
}
```

Jak widać z powyższych listingów część wartości jest wpisana na sztywno i wkrótce zostanie podmieniona na te bardziej poprawne zgadzające się z samą grą. Obecne są jedynie dla testów.

W omawianej wersji kodu widzimy analogiczną inicjalizację do tej z oryginalnego kodu, a sama kolejność wykonania poszczególnych instrukcji jest zachowana, np:

listing.bas

```
20    init_values()
    clr_scrn()
    set_border_color(BLACK)
    set_inner_color(RED)
    prepare_consts_to_load()
```

Ma odpowiednika w C++ (z dodatkową kontrolą błędów):

Game.cpp C++

```cpp
int Game::Run() {
    /* Init section (original) */
    if(init_values_()) {
        return 1;
    }

    rectangle_screen_.setOutlineColor(c64::BLACK);
    rectangle_screen_.setFillColor(c64::RED);
    color_clear_ = DEFAULT_CLEAR_COLOR;

    if (prepare_consts_to_load_()) {
        return 2;
    }
```

W obecnej wersji kod wchodzi do menu skąd jesteśmy w stanie wyjść i zobaczyć jedynie ekran z ramką.

Game.cpp C++

```
77 int Game::show_menu_() {
78     bool exit = false;
79     while(!exit)
80     {
81         while(window_.pollEvent(event_)) {
82             if (is_exit_triggered_())
83                 exit = true;
84         }
85 
86         // TODO menu stuff (players selection, etc...)
87         clear_screen_();
88         window_.draw(rectangle_screen_);
89         draw_logo_();
90         window_.display();
91     }
92 
93     return 0;
94 }
```

W najbliższej wersji chciałbym stworzyć w miarę wierną kopię menu z gry wraz z jego pełną funkcjonalnością, a więc ze wszystkimi elementami poprzedzającymi właściwą rozgrywkę.

Podsumowanie

Od ostatniego momentu nieco się podziało z projektem, na koniec chciałbym przypomnieć jeszcze raz o swojej prośbie odnośnie tłumaczeń tekstów (jeżeli jesteście sami gotowi przetłumaczyć teksty lub znacie kogoś takiego to dajcie znać).

Tradycyjnie zapraszam Was do komentowania, a także śledzenia bloga przez fb, Twittera, G+ (linki do *Social Media* są w panelu po prawej). Jeżeli uważacie, że komuś z Waszych znajomych może spodobać się ten artykuł to oczywiście zachęcam do podzielenia się z nim linkiem do bloga ;)

W kolejnym wpisie z tej serii mam nadzieję przedstawić Wam przynajmniej w pełni działające menu, także do przeczytania i..

Code ON!