

**Autor: Daniel "zoNE" Gabryś**  
e-mail: o2code@interia.pl  
**Wszelkie prawa zastrzeżone.**

# **Jak tworzyć Mody w Edytorze Map "Original War".**

**Wersja Kursu: 1.1**  
**Zgodność z wersją Edytora: 2.2 lub nowszą**

# Spis Treści:

1. Co nam daje Edytor? .....	3
2. Interfejs Programu .....	4-6
3. Interfejs – Okno Edycji .....	7-11
a. Paski Narzędzi .....	7-9
b. Menu Programu .....	9-11
4. Jak Uruchomić Program? .....	12
5. Początki Z Programem, Czyli Jak, Co I Gdzie (FAQ)... ..	13-17
6. Bardziej zaawansowane opcje .....	18-29
a. Postacie, Budynki, Pojazdy .....	18-22
b. Dodawanie Drzew, Kamieni, itd. ....	23
c. Dodawanie Trawy .....	24
d. Formowanie terenu .....	25
e. Ustawienia mapy, oraz Dobieranie Sojuszy .....	26-27
f. Dostępność budynków w misjach .....	28
g. Dostępność technologii w misjach .....	29
7. Skróty Klawiaturowe .....	30-31
8. Sail, Jako Język Skryptowy .....	32-90
a. Jak uruchomić Sail Editor? .....	32-33
b. O Sail Edytorze słów kilka .....	34
c. Składnia Saila .....	35-36
d. Stałe Saila .....	37-45
e. Funkcje Saila .....	46-90
9. Zakończenie .....	91

## \* Co nam daje Edytor?

Zanim zaczniemy programować w *Edytorze Map "Original War"*, może najpierw zastanówmy się *po co nam on?*

*Edytor* sam w sobie, jako program jest dosyć prosty w obsłudze, problemy jednak zaczynają się gdzie indziej... Problemy zaczynają się, gdy zaczynamy do edycji naszych map używać funkcji *SAIL'a*...

<p><b>SAIL</b> - język skryptowy gry oparty na języku programowania <i>Delphi (Pascal)</i>, sama składnia języka nie różni się wiele, od standardowego <i>Delphi</i>, lecz różnią się w nim tzw. Komendy. <b>Delphi</b> - język programowania oparty na <i>Object Pascalu</i>, w którym zostało napisane "<i>Original War</i>".</p>
---

Kurs ten dokładnie opisuje funkcje programu "*Original War Editor*", oraz zastosowanie w nim funkcji *SAIL'a*, żeby bardziej już nie zanudzać Państwa, przejdźmy już do konkretów, czyli następnego działu ;-).

## \* Interfejs Programu

Aby nauczyć się programować w *Edytorze*, trzeba najpierw poznać jego interfejs. W przypadku *Edytora* w wersji 2.2, oraz nowszej, interfejs jest intuicyjny, oraz prosty w obsłudze, więc nie powinno być z tym większych problemów.

Gdy włączymy program, naszym oczom ukaże się taki panel startowy, jaki jest przedstawiony na rysunku 2.1, w którym wybieramy jaki *Mod* chcemy edytować.



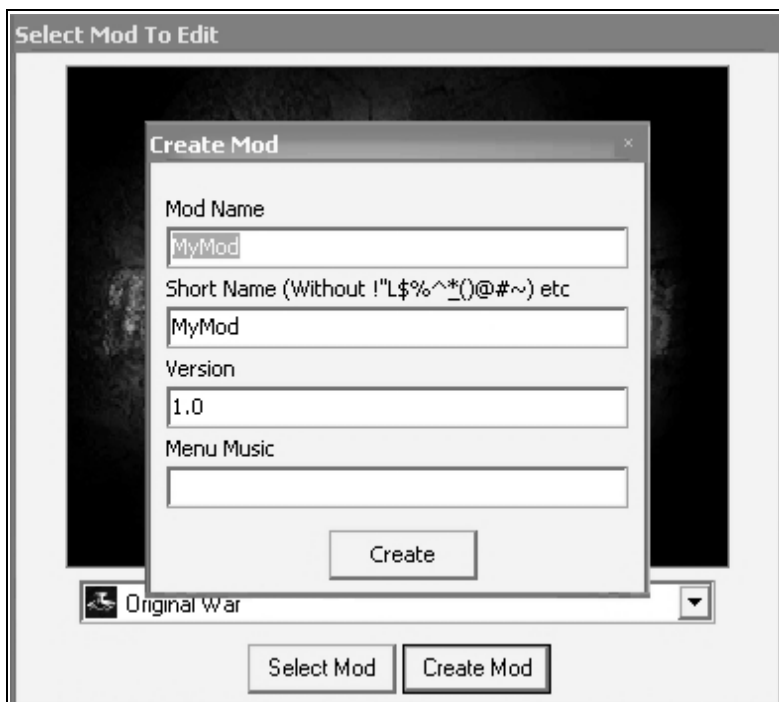
Rysunek 2.1 – Okno widoczne zaraz po uruchomieniu *Edytora*.

Jak to widać na rysunku 2.1, w pierwszym oknie, jakie widzimy po uruchomieniu programu, są widoczne dwa przyciski, 'Select Mod', oraz 'Create Mod'.

**Select Mod** – po wybraniu z listy odpowiedniego moda, jaki chcemy edytować, naciskamy ten przycisk, aby potwierdzić.

**Create Mod** – naciskamy ten przycisk, gdy chcemy utworzyć nowy Mod, a nie edytować stary.

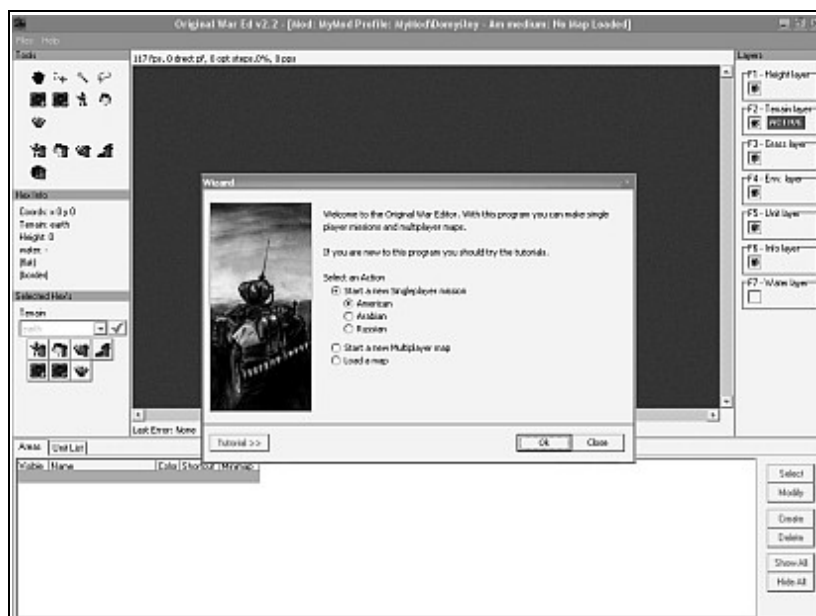
Po naciśnięciu przycisku 'Create Mod' pojawi się okno widoczne na rysunku 2.2.



Rysunek 2.2 – Okno widoczne po naciśnięciu przycisku 'Create Mod'.

W tym oknie mamy za zadanie wpisać kolejno nazwę Modu, skróconą nazwę Modu (nazwę pliku czyli nie używamy znaków typu: !@#\$\$%^&\*()\_~", itd.), oraz wersję Modu.

Gdy już wybierzemy, czy chcemy utworzyć nowy Mod, lub edytować stary, to ukaże się przed naszymi oczyma główne okno programu, widoczne na rysunku 2.3.



Rysunek 2.3 – Główne okno programu.

W tym oknie mamy widoczne opcje wyboru czy ma to być nowa mapa do kampanii, trybu multiplayer, lub czy ma wczytać wcześniej zapisaną przez nas mapę, w przypadku mapy do kampanii mamy do wyboru, też nację tej kampanii, czyli Amerykanie, Rosjanie, oraz Arabowie.

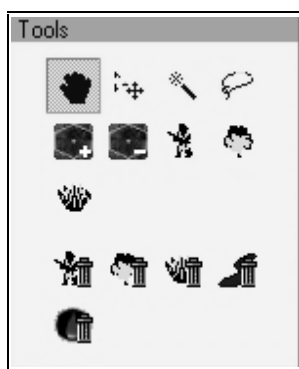
Gdy już wybierzemy jaki to ma być rodzaj mapy, ujawnią się nam opcje edycyjne edytora, które dokładniej opiszę w następnym dziale zatytułowanym 'Interfejs – Okno Edycji'. A więc zapraszam do następnego działu ;-).

## \* Interfejs – Okno Edycji

W tym dziale opiszę, jak jest zbudowane okno edycji, w *Edytorze Map*, a więc idziemy po kolei tak jak to jest w programie...

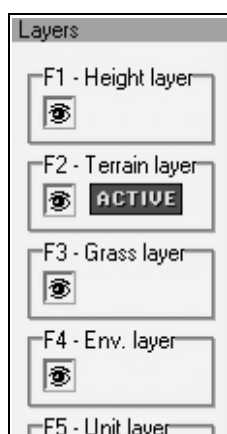
### 1. Paski Narzędzi:

- Pierwszym i chyba najważniejszym paskiem jaki spotkamy tutaj, jest pasek "Narzędzia", ten pasek, będzie nam przydatny do wykonywania najważniejszych funkcji, między innymi takich jak: dodawanie do mapy pojazdów, budynków, ludzi, zaznaczanie terenu, dodawanie drzew, itd. Pasek widoczny na rysunku 3.1.1.



Rysunek 3.1.1 – Paski Narzędzi – Narzędzia.

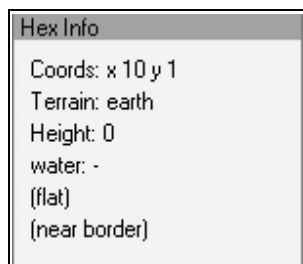
- Drugim równie ważnym paskiem, jest pasek "Warstw", za jego pomocą możemy zaznaczyć, czy dana warstwa ma być aktywna, widoczna, itd. Widoczny jest on na rysunku 3.1.2.



Rysunek 3.1.2 – Paski Narzędzi – Warstwy.

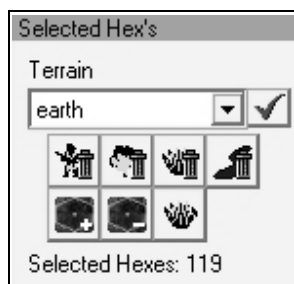
- Kolejnym paskiem jaki spotkamy w *Edytorze*, będzie pasek "Hex Info", który wskazuje nam w

jakim miejscu jest nasz kursor myszy (na osiach x,y), pokazuje nam jakie w tym miejscu jest podłoże (ziemia, kamienie, itd.), czy w danym miejscu jest woda, oraz czy przypadkowo nie wyszliśmy kursorem, poza obszar mapy. Pasek widoczny na rysunku 3.1.3.



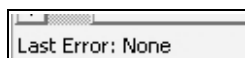
**Rysunek 3.1.3** – Paski Narzędzi – Hex Info.

- Następny pasek to pasek "Zaznaczenia Hex", ten pasek zezwala nam na zmianę zaznaczonego terenu (np. ziemia, skały, itd.), pokazuje nam też jaki fragment terenu aktualnie jest zaznaczony. Pasek widoczny na rysunku 3.1.4.



**Rysunek 3.1.4** – Paski Narzędzi – Zaznaczenie Hex

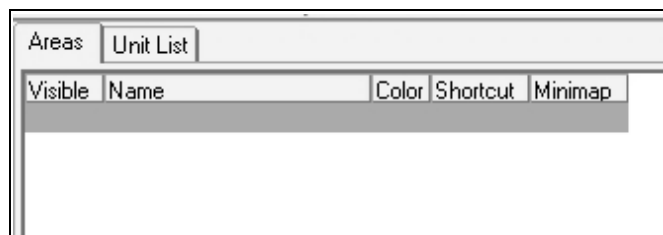
- Są jeszcze inne paski, które posiadają inne przydatne funkcje, jednym z nich jest np. pasek błędów, jest on umieszczony pod okienkiem mapy i jest na nim wyświetlana wszelkie błędy jakie mogą się pojawić w czasie tworzenia mapy, np. że podwozie jest za lekkie, lub za ciężkie dla danego silnika, itd. Patrz rysunek 3.1.5.



**Rysunek 3.1.5** – Paski Narzędzi – Błędy

- Ostatnim z pasków i jakże ważnym paskiem jest pasek główny, w którym po wybraniu danej opcji z poprzednich pasków, pojawiają się zakładki, np. po naciśnięciu przycisku stawiania postaci na pasku głównym pojawi się zakładka, w której wybieramy, jakiej ma być nacji dana postać, czy ma być to postać, czołg, czy może budynek, itp. Widoczny na rysunku 3.1.6.





Rysunek 3.1.6 – Paski Narzędzi – Pasek Główny

## 2. Menu programu:

W tej pod kategorii zajmiemy się menu głównym w programie, widocznym na samej górze programu, menu jest podzielone na początku na dwie kategorie, po wybraniu opcji edycji mapy menu zawiera już czternaście kategorii, są to kolejno: Pliki, Testowanie, Tryby, Edycja, Zaznaczenie, Narzędzia, Warstwy, Obszar, Modyfikacja, Wsparcie Sail'a, Widok, Okna, Misje, oraz Pomoc, każda z tych kategorii ma przypisane inne możliwości, ale o tym za chwilę, Fragment paska jest widoczny na rysunku 3.2.1.



Rysunek 3.2.1 – Menu Programu

A teraz po kolei opiszę co oznaczają dane przyciski w menu, ostrzegam, iż w wersjach innych, niż 2.2 mogą być małe różnice w budowie menu, a więc lecimy z tym:

### **1. Menu Pliki (Files):**

- Open - Otworz
- Restart - Restart
- Wizard - Kreator
- Save - Zapisz
- Save As - Zapisz jako
- Save a Copy - Zapisz kopie
- Map Preferences - Ustawienia mapy
- Editor Preferences - Ustawienia edytora
- Resize Map - Zmiana wielkości mapy
- Export Ground Bitmap - Eksport bitmapy podłoża
- Export Bitmaps - Eksport bitmapy
- Export Water Height Map - Eksport mapy wysokości wody
- Export Height Map - Eksport wysokości mapy
- Export Water Images as bin - Eksport obrazu wody jako bin
- Export Ground Env - Eksport podłoża env
- Export Map Visual - Eksport wizualny mapy
- Export Vehicles - Eksport pojazdów
- Import Bitmap - Import bitmapy
- Import Water Height Map - Import wysokości wody mapy
- Import Water Anim Map - Import animacji wody mapy
- Import Water Image as bin - Importuj obrazy wody jako bin
- Import GBI - Import GBI
- Exit OW - Wyjdź z OW

### **2. Menu Testowanie (Testing):**

- Difficulty - Trudność
  - >> • Adventure (Easy) - Przygoda (łatwy)
  - >> • Commander (Mediom) - Dowódca (średni)
  - >> • Master Strategist (Hard) - Mistrz strategii (trudny)
- Campaign - Kampania
  - >> • American – Amerykanie
  - >> • Arabian – Arabowie
  - >> • Russian – Rosjanie
- Profile - Profil
- Open Campaign Save - Otwórz zapis kampanii

### **3. Menu Tryby (Mode):**

- Go To Game Mode - Przejdź do trybu gry

### **4. Menu Edycja (Edit):**

- Undo – Cofnij
- Redo – Przywróć

### **5. Menu Zaznaczenie (Selection):**

- All - Wszystko
- None - Nic
- Invert - Przeciwnie
- Hide - Ukryj
- Expand - Rozszerz
- Contract - Skurcz
- Set Unreachable - Ustawić niedostępne
- Set Reachable - Ustawić dostępne

### **6. Menu Narzędzia (Tools):**

- Hand Selection - Ręczna selekcja
- Lasso Selection - Lasso selekcja
- Move Selection - Ruch selekcja
- Magic Wand Selection - Magiczna różdżka selekcja
- Create Unit - Stwórz jednostkę
- Create Enviroment - Stwórz środowisko
- Grass Tool - Narzędzia trawy

### **7. Menu Warstwy (Layers):**

- Height Layer - Warstwa Wysokości
- Terrain Layer - Warstwa Terenu
- Grass Layer - Warstwa Trawy
- Enviroment Layer - Warstwa Środowiska
- Unit Layer - Warstwa Jednostki
- Info Layer - Warstwa Info
- Water Layer - Warstwa Wody
- Height Layer Visibility- Warstwa wysokości widoczna
- Terrain Layer Visibility - Warstwa terenu widoczna
- Grass Layer Visibility - Warstwa trawy widoczna
- Enviroment Layer Visibility - Warstwa środowiska widoczna
- Unit Layer Visibility - Warstwa jednostki widoczna
- Info Layer Visibility - Warstwa info widoczna
- Water Layer Visibility - Warstwa wody widoczna
- Show All - Pokaz wszystkie

### **8. Menu Obszar (Areas):**

- Create New Area - Stworzyć nowy obszar
- Delete Area - Usunąć obszar
- Select Area - Zaznaczyć obszar

- Add Area to selection - Dodać obszar do zaznaczenia
- Sub Area From Selection - Podłożyć obszar z zaznaczenia
- Combine Area With Selection - Zestawić obszar z zaznaczenia
- Redefine Area - Rozpoznać obszar
- Add Selection To Area - Dodać zaznaczenie do obszaru
- Sub Selection From Area - Podłożyć zaznaczenie z obszaru
- Combine Selection With Area - Zestawić zaznaczenie obszaru
- Show All Areas - Pokazać wszystkie obszary
- Hide All Areas - Ukryć wszystkie obszary

### **9. Menu Modyfikacja (Modify):**

- Apply Terraforming - Zatwierdzić formowanie terenu
- Remove Border Grass - Usunąć granicę trawy
- Compute Water Zbuffer - Komputerowy zbuffer wody
- Reset FoW - Reset FoW
- Reset FoW Grass - Reset FoW trawy
- Reset LoS - Reset LoS
- Reset Faces - Reset twarzy
- Update hexes - Update hexów

### **10. Menu Wsparcie Sail'a (Sail Support):**

- Base Position - Pozycja bazy
- Set Of Points On - Ustaw punkty na

### **11. Menu Widok (View):**

- Grid Visible - Siatka widoczna
- Dummy Ground Mode - Tryb sztucznego podłoża
- Render Terrain - Rozdzielacz terenu
- Disable GBI - Wyłącz GBI
- Z-Buffer View - Z-Buffer widoku
- Toggle FoW All Visible - Przełącz FoW wszystkie widoczne
- Fog Recount Mask - Mgła odpowiednik maskujący
- Show All Occupied - Pokaż wszystkie zajęte
- Show All Working - Pokaż wszystkie działające

### **12. Menu Okna (Windows):**

- Sail Editor Visible - Widoczny Sail Edytor
- Grass Filter - Filtr trawy
- Inter. Configs Visible - Widoczne ustawienia wewnętrzne
- Develop Visible - Widoczne rozwijanie
- Minimap Visible - Widoczna minimapa
- Memory Status - Stan pamięci
- Debug Visible - Widoczne błędy

### **13. Menu Misje (Mission):**

- Attitudes - Stanowiska
- Technologies - Technologie
- Restrictions – Ograniczenia

### **14. Menu Pomoc (Help):**

- About – O...

## \* Jak Uruchomić Program?

Sprawa z uruchamianiem programu czasami bywa kłopotliwa, gdyż nie każdy wie, iż edytor ma swoje wymagania, co do ustawień systemowych systemu Windows. A więc aby nie przeciągać, przejdźmy do szczegółów, edytor wymaga ustawienia rozdzielczości ekranu 1024x768, oraz jakości kolorów 16 bitów. Jak to zrobić? A więc po kolei:

1. Klikamy prawym przyciskiem myszy na pulpit.
2. Pojawi się menu, z którego wybieramy opcję '*Właściwości*', zazwyczaj ostatnia...
3. W nowo otwartym oknie wybieramy zakładkę '*Ustawienia*'.
4. Następnie w tabeli '*Rozdzielczość Ekranu*' wybieramy rozdzielczość 1024x768.
5. W tej samej zakładce, w tabeli '*Jakość Kolorów*' wybieramy opcję 16 bitów.
6. Klikamy przycisk '*Zastosuj*'.
7. Pojawi się okno z zapytaniem czy na pewno chcesz zmienić te ustawienia, zatwierdzamy.
8. Oraz klikamy '*OK*', w oknie właściwości.

Gdy już tak ustawimy nasz system, wtedy bez problemów powinien się uruchomić *Edytor*, a i zapomnielibym, jak w czasie uruchamiania *Edytora* wyskoczy okienko z prośbą włożenia do napędu płyty CD, wtedy klikamy na '*Anuluj*', i wszystko będzie dobrze.

# \* Początki Z Programem, Czyli Jak, Co I Gdzie (FAQ)...

A więc doszliśmy już do działu, w którym dokładnie opiszę, jak co się robi w tym *Edytorze*. Ten dział będzie zrobiony bardziej w stylu takiego małego FAQ, w którym będą umieszczone najczęściej zadawane przez ludzi pytania, oraz odpowiedzi na nie...

## 1. Wybrałem już jednostkę, budynek, lub chcę zmienić ukształtowanie terenu, klikam i nic się nie dzieje... Co mam zrobić?

Odp.: Aby jakieś zadanie typu położenie budynku na mapie, itd., było wykonane musimy najpierw nacisnąć na klawiaturze przycisk *CTRL+SHIFT*, oraz niezwalniając go kliknąć *lewym przyciskiem myszy*, w dane miejsce na mapie, w którym chcemy dokonać danego upuszczenia.

## 2. Jak dodać dialogi do misji?

Odp.: Pliki dialogu są umieszczone w *Campaigns\%STRONA%* (Gdzie % STRONA % jest Am albo Ru) i nosi nazwę *Txt%MID%.wri* (Gdzie %MID% to numer misji, *i.e 01*). To plik historii wszystkich tekstów w dialogu, wszystkie pytania w grze, oraz cele misji.

Dialog wygląda podobnie jak poniższy:

```
// Dialog B - nazwa kategorii
$ VelNow
- Teraz! Dostańmy ich!
$ Nazwa2
- dialog...2
$ Nazwa3
- dialog...3
```

Jeśli chcesz użyć dźwięku dla dialogów nazwy zbioru muszą być *%MOD%\sound\dialogs\%SIDE%\%MID%\%DNAME%.wav* (Gdzie %DNAME% jest nazwą dialogu z usuniętą pierwszą częścią i dodanym %MID%).

Dla przykładu:

```
$ VelNow
- Teraz! Dostańmy ich!
```

Byłby *%MOD%\sound\dialogs\ru\01\01\_VelNow.wav*

Dialog \*.audio format wav musi mieć PCM 22.050 kHz, 16 Bit, Mono.

## 3. Jak dodawać misje do moda?

Odp.: Każda strona ma plik *missions.dat*, który zawiera listę misji, który mamy przejść (twój

ogranicznik to 30 misji). Te są mieszane w `\Campaigns\%SIDE%\` (gdzie `%SIDE%` jest Am albo Ru).

**Ważne wyrazy:**

MAP - nazwa mapy (folderu z misją)

NAME - nazwa, która jest widoczna na ekranie wczytywania się misji

SUBCAMP - od 1 do 5, kontroluje obrazek pokazywany po zwycięstwie

PREV - numer poprzedniej misji

NEXT - numer kolejnej misji

NEXTQ - (aktualnie nieznana opcja)

FINISH - oznacza że gra kończy się po ukończeniu tej misji

VIDEO 1 - oznacza, iż będzie pokazywane wideo

VIDEONAME - nazwa wideo do załadowania (z `%MOD%\Videos\`), przedzająca konieczność użycia VIDEO 1

Dla przykładu wycinki z pliku `missions.dat`, z *Test Moda by Stucuk*:

```
CAMPAIGN "Russian campaign"
MISSION 0
  NAME "New Campaign"
  NEXT 1
MISSION 1
  MAP Ar01
  NAME "Something Went Wrong"
  SUBCAMP 1
  PREV 0
  NEXT 2
  VIDEO 1
  VIDEONAME ar01.seq
MISSION 2
  MAP 02
  NAME "Hunting Grounds"
  SUBCAMP 1
  PREV 1
  NEXT 3

MISSION 11
  MAP 11
  NAME "New end"
  SUBCAMP 3
  PREV 10
  NEXTQ "" 3 "" 20 "" 12 "" 16

MISSION 15
  MAP 15
  NAME "Final Battle"
  SUBCAMP 4
  PREV 14
  FINISH
END_OF_CAMPAIGN
```

#### 4. Jaki program służy do edytowania, oraz tworzenia plików video (intro do misji, itd.) do OW?

Odp.: Filmiki do OW (czyt. intra, itd.) tworzy się w programie *PRCiX*, który jest dostępny razem z *patchami od 1.03* wzwyż.

#### 5. Jak edytować pliki postaci, budynków, itd., które są zapisane w formacie \*.gms, oraz \*.gmz, czy to jest możliwe?

Odp.: Niestety nie da się edytować wyglądu postaci aktualnie, lecz da się je przeglądać, w programie *gm\_view.exe*, który jest dostępny w *patchu 1.03*, oraz nowszych.

#### 6. Jak dodać zegarek do mapy?

Odp.: Wklej do *Sail* kodu mapy poniższy kod:

```
every 0$1+0$0.5 do
begin
display_strings=[#Multi1x1-Time',tick];
enable;
end;
```

#### 7. Jak korzystać z *GrndSec*?

Odp.: Rób po kolei:

1. Uruchom *ow\_editor*.
2. Otwórz swoją mapę.
3. Wybierz w menu **Files -> Export Ground Bitmap**, zapisz mapę.
4. Zamknij *ow\_editor*.
5. Zmodyfikuj bitmapę, którą zapisałeś.
6. Uruchom *grndsec*.
7. Wybierz z menu **File -> Load\_BMP** (wybierz bitmapę, którą zmodyfikowałeś).
8. Kliknij *Autopack*.
9. A następnie wybierz **GBI -> Save**.
10. Zamknij *grndsec*.
11. Uruchom *ow\_editor*.
12. Otwórz swoją mapę.
13. Wybierz menu **Files -> Import GBI**.
14. Potem kliknij menu **Files -> Save**.
15. Oraz wybierz menu **Files -> Restart**.

Twoja ziemia powinna teraz zostać pokazana.

#### 8. Jak to zrobić, aby postacie siedziały w budynkach?

Odp.: Nic prostszego. Stawiasz na mapie jakąś postać, oraz budynek, potem wybierasz z menu **Mode -> Go To Game Mode**, a następnie w trybie gry wkładasz te postacie do budynków, klikasz *ESC*, a następnie wybierasz opcję "Wróć do Edytora", z menu które się pojawi. To by było na tyle.

**UWAGA:** Nie poleca się używania tego sposobu, lepiej zrobić to przy pomocy *SAIL'a*.

### 9. Jak wyznaczyć, aby ludzie z twojego teamu, lub innego, robili coś samodzielnie, np. budowali budynki, chodzili gdzieś, itd.?

Odp.: Teoretycznie mogło by się to wydawać trudne, lecz w praktyce, jest to bardzo proste. A więc kolejno:

- Stawiasz na mapie postać (ta która ma coś sama robić).
- Wybierasz z menu programu **Mode -> Go To Game Mode**.
- Gdy uruchomi się gra wciskasz **Pauzę**.
- A następnie wybierasz postać (tą która ma coś robić), oraz planujesz jej zadanie, tak jak byś to robił w grze, czyli np. klikasz na nią, przytrzymujesz **SHIFT**, naciskasz w kilka miejsc, w które ma iść, każesz jej coś gdzieś zbudować.
- Potem niewyłączając spacji, klikamy **ESC**, oraz wybieramy z menu, które się pojawi opcję "Wróć do Edytora".
- Zapisujemy zmiany.

No i to by było na tyle, po uruchomieniu modu, postaci, którym zadałeś jakieś zadania, będą same się poruszały (oczywiście jeżeli będą z twojego teamu, to będziesz też mógł nimi ruszać).

**UWAGA:** Nie poleca się używania tego sposobu, lepiej zrobić to przy pomocy SAIL'a.

### 10. Jak dodać własną muzykę do gry?

Odp.: Trzeba umieścić w folderze z modem folder **Sound**, do którego należy umieścić pliki dźwiękowe o takich samych nazwach jak są one oryginalnie nazwane w grze (są to pliki o rozszerzeniu **\*.wav**).

Np. by zmienić muzykę z menu głównego, musisz dać plik do folderu **Sound\Hudba\**, o nazwie **Menu.wav**, z rozszerzeniem **\*.wav** oczywiście.

Nazwy plików możesz sprawdzić w pliku **Data2.owp**, w folderze z grą, a przesłuchać je możesz po ściągnięciu ich (jako soundtracki) np. z oficjalnej strony OW.

### 11. Co zrobić, aby usunąć jakiś przycisk z głównego menu gry (np. przycisk **Gra wieloosobowa**)?

Odp.: Jest to bardzo proste. Za wyświetlanie przycisków w menu odpowiada plik **UIDlgAlien.txt** i to właśnie ten plik będziemy edytować (znajduje się on w pliku **Data1.owp**).

A więc będziemy tu działać, na przykładzie przycisku "Gra wieloosobowa". Po otwarciu pliku w programie Notatnik (Notepad), szukamy fragmentu **[ButtonFlow multiplayer]**:

```
[ButtonFlow multiplayer]
15 103
0
"main_inactive"
"Button text"
[end]
```

Umieszczone pod nim dane odpowiadają za przycisk "Gra wieloosobowa". Tj. umiejscowienie na osi x, y, rodzaj grafiki, oraz kolor czcionki.



Aby usunąć takowy przycisk, wystarczy jedynie wpisać komendę *NotEnabled*, przed *[end]* i przycisk zostanie automatycznie wyłączony.

Powinno to wyglądać tak:

```
[ButtonFlow multiplayer]
15 103
0
"main_inactive"
"Button text"
NotEnabled
[end]
```

## 12. Jak zmienić napis *Wersja Gry: ...*, na *Wersja Moda: ...*:

Odp.: Za wyświetlanie tego napisu odpowiadają pliki językowe *Lang<język>.wri* umieszczone w folderze *Texts*, w pliku *Data1.owp*. Wyszukujemy tam linijkę:

```
1024=Wersja Gry: <VER>
```

W której wystarczy zmienić *Wersja Gry: <VER>*, na *Wersja Moda: 1.0*, lub jakiś inny napis.

Powinno to tak wyglądać:

```
1024=Wersja Moda: 1.0
```

## 13. Co trzeba zrobić, aby po kliknięciu przycisku *Autorzy*, pojawiła się lista autorów moda?

Odp.: Odpowiadają za to dwa pliki *tit1.bin*, oraz *tit2.bin*, są one umieszczone w folderze *Titles*, w pliku *Data1.owp*.

Aby zmienić ich zawartość, należy stworzyć po prostu nowe programem *TitleCreator*, który znajduje się w folderze *Original War\Utils>TitleCreator*. Po stworzeniu nowych plików, należy je przekopiować do folderu *Titles*, w folderze z twoim modem.

Oto znaczniki, jakich możemy używać tworząc listę autorów:

- \* Tytuł (duże litery)
- + Kategoria, pod którą wypisujemy osoby (np. Grafika)
- # Ksywka (np. zoNE)
- / Opis

Przykład:

```
* Tytuł

+ Grafika
Daniel G # zoNE

/To jest mój pierwszy mod
```

## \* Bardziej zaawansowane opcje

W tym rozdziale opiszę, jak korzystać z bardziej zaawansowanych opcji, w *Edytorze*.

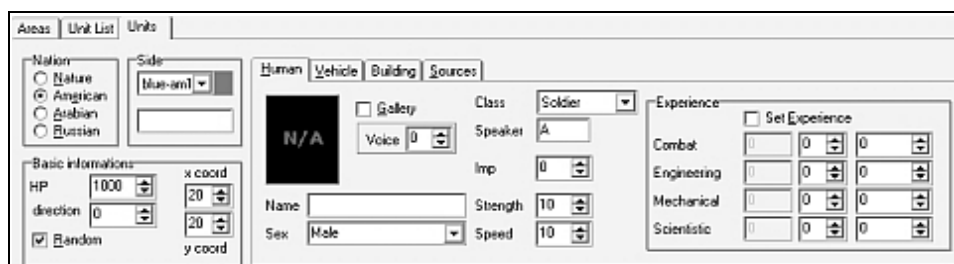
### 1a. Postacie, Budynki, Pojazdy:

A więc krótki opis, jak stawiać na mapie postacie, budynki, oraz pojazdy.

Aby włączyć menu wyboru "Units", w głównym pasku narzędzi, najpierw musimy wybrać ikonkę:



Następnie, po jej naciśnięciu pojawi się na dole programu zakładka "Units", wyglądająca tak, jak na rysunku 6.1.1.



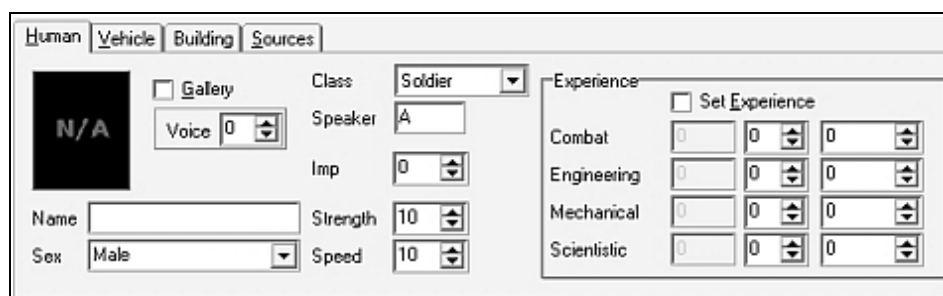
Rysunek 6.1.1 – Zakładka Units

W tej zakładce mamy takie tabele jak:

- Nation, w której wybieramy jakiej nacji ma być to co chcemy dać na mapę. Mamy tu do wyboru Nature (bez strony), American, Arabian, oraz Russian.
- Side, w której wybieramy kolor temu, do którego ma to coś co chcemy dać na mapę należeć.
- Basic Informations, ostatnia tabela, w której wybieramy ile ma mieć HP (życia) nasz obiekt, oraz w którym miejscu ma być umieszczony (osie x, y) nie jest ta opcja konieczna, gdy używamy skrótu SHIFT + PPM w miejscu gdzie chcemy postawić obiekt, wtedy osie x i y wybiorą się automatycznie.

Są w niej też dostępne dodatkowe zakładki tj. Human (Postać), Vehicle (Pojazd), Building (Budynek), oraz Sources (Surowce), służą one do wyboru czym ma być obiekt, który chcemy umieścić na mapie, czy ma być to postać, pojazd, budynek, czy surowiec (tj. ropa, syberyty, itd.). Poniżej znajduje się ich dokładny opis.

a. No więc zajmijmy się najpierw Human (Postać), rysunek 6.1.2.



**Rysunek 6.1.2** – Zakładka Human

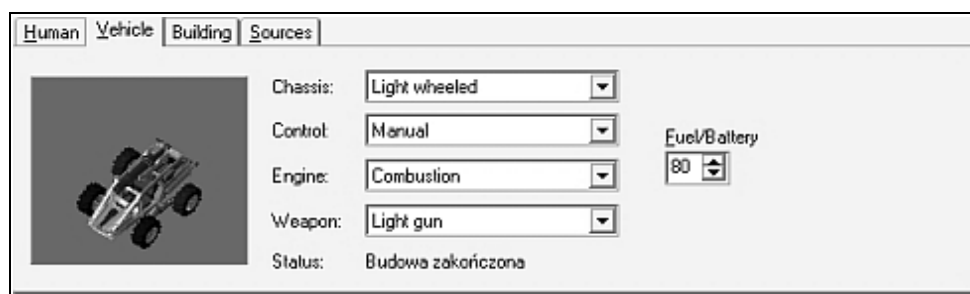
Co mamy tu do dyspozycji? Mianowicie mamy tu kilka opcji dotyczących postaci. Są nimi:

- Gallery (wybierz losowe zdjęcie postaci z galeri)
- Voice (zamiast wybierać losowe zdjęcie, można wpisać numerkę danej fotki z tych z kampanii, i ma się już postać np. MacMillana, w tabeli poniżej jest opisane kilka najważniejszych postaci (nie jestem pewien czy to dokładnie te numery bo dawno z tego nie korzystałem, ale powinny się zgadzać:

1 - Burlak
8 - Płk. Iwan P. Kurin
17 - Mar. Yashin
18 - Brat Burlaka
48 - Joan (dziew. MacMillana)
59 - Pobity Burlak
60 - Maj. D. N. Płatonow

- Name (tu piszemy, jak ma się nazywać nasza postać, np. MacMillan).
- Sex (tu wybieramy płeć, np. Male (Mężczyzna), Female (Kobieta).
- Class (tu wybieramy jakiej rangi ma być to postać, np. Soldier (Żołnierz), Engineer (Inżynier), Mechanic (Mechanik), itd.).
- Imp (tu wybieramy, czy dana postać ma być przywódcą grupy czy nie, tak jak np. MacMillan, czyli 110).
- Speaker (tu możemy wybrać od A do Z, jakim głosem, ma przemawiać nasza postać).
- Speed (ta opcja służy do wyboru, z jaką prędkością będzie się poruszała nasza postać, po mapie).
- Experience (ustawienia umiejętności postaci, mamy tu do wyboru: Combat (walka), Engineering (budowanie), Mechanical (mechanizacja), Scientific (naukowe).

b. Następną zakładką to Vehicle (Pojazdy), rysunek 6.1.3.

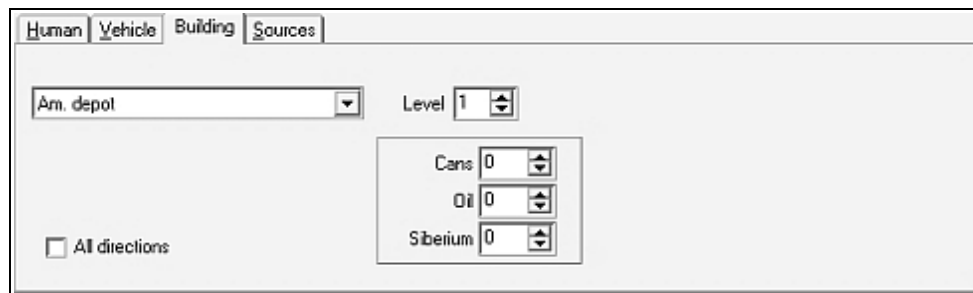


**Rysunek 6.1.3** – Zakładka Vehicle

Jest w niej dostępna niewielka ilość opcji tj.:

- Chassis (Podwozie, wybierasz tu na jakim podwoziu ma być twój sprzęt).
- Control (Sterowanie, tu wybierasz, kto ma sterować pojazdem, czy ma być to człowiek, komputer, itd.).
- Engine (Silnik, wybieramy jakiego gatunku ma być silnik, czy ma być to silnik syberytowy, na ropę, itd.).
- Weapon (Uzbrojenie, tu z kolei wybieramy jak nasze cacko ma być uzbrojone, ciężkie działo, karabin maszynowy, itd.).
- Status (tutaj możemy zobaczyć czy dane części pasują do siebie, czy silnik nie jest za ciężki względem podwozia, itd.).
- Fuel/Battery (Paliwo/Bateria, tu wybieramy w jakim stopniu ma być napełnione zasilanie w naszym wozie, np. bateria w zasilaniu słonecznym).

c. Kolejna zakładka to Building (Budynek), rysunek 6.1.4.

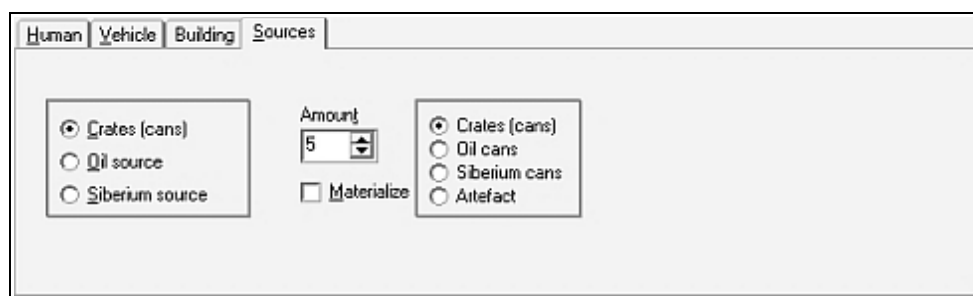


**Rysunek 6.1.4 – Zakładka Building**

Zawiera ona opcje tj.:

- Umieszczony w lewym górnym rogu pasek wyboru jaki to ma być budynek (np. skład, laboratorium, lub rozbudowanie dla fabryki).
- Level (poziom budynku, od 1 do 10)
- Opcje specjalne budynku (np. w przypadku składu tj. ilość Cans (skrzynek), Oil (bężyny), Siberium (syberytu), w składzie).

d. No i w końcu ostatnia zakładka Sources (Surowce), rysunek 6.1.5.



**Rysunek 6.1.5 – Zakładka Sources**

W tej ostatniej już zakładce, mamy do dyspozycji opcje tj.:

- Opcje wyboru surowca, jaki ma być umieszczony na mapie (np. Crates (Skrzynki), Oil source (Olej), Siberium source (Syberyt)).
- Oraz opcje dodatkowe danego surowca (w przypadku skrzynek np. Amount (ilość), oraz opcje wyboru jaki ma być surowiec w skrzynkach, np. Crates (Skrzynki), Oil cans (Olej), Siberium cans (Syberyt), Artefact (Artefakt)).

### **1b. Usuwanie Postaci, Budynków, Pojazdów, oraz Surowców:**

Aby usunąć jakiś obiekt, należy najpierw wybrać z paska narzędzi "Tools", ikonkę:



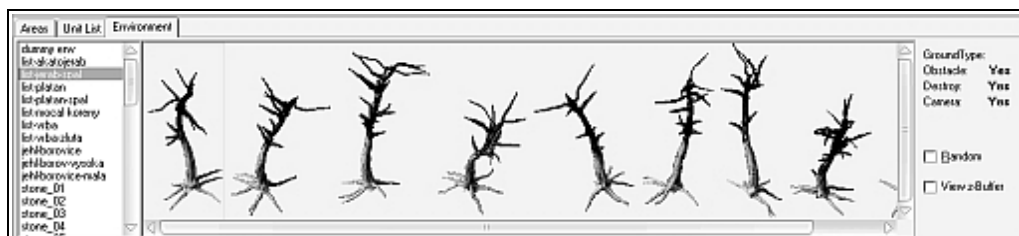
a następnie przytrzymać przycisk *SHIFT*, oraz kliknąć *PPM*, na obiekt, który chcemy usunąć.

## 2a. Dodawanie Drzew, Kamieni, itd.:

Dodawanie drzew, kamieni i innych tekstur do mapy, jest dosyć proste. Najpierw z paska narzędzi "Tools", wybieramy ikonkę:



następnie pojawi się na dole zakładka "Environment", spójrz na rysunek 6.2.



Rysunek 6.2 – Zakładka Environment

W tej zakładce wybieramy opcje, jaki to ma być obiekt, oraz po wybraniu już, przytrzymujemy *SHIFT*, oraz klikamy *PPM*, w miejsce, w którym ma się znaleźć dany obiekt.

Oto spis dostępnych tu opcji:

- Tabela wyboru kategorii tekstury (tu wybieramy jakiej ma to być kategorii tekstura, np. zeschnięte drzewo, itd.).
- Okno wyboru tekstury (po wybraniu kategorii, właśnie w tym oknie wyświetli się lista tekstur zawartych, w tej kategorii).
- GroundType (opis, na jakich podłożach, może być stosowana dana tekstura).
- Random (losowa tekstura).
- View z-Buffer (widok z-Buffera)

## 2b. Usuwanie Drzew, Kamieni, itd.:

Usuwanie jest podobne, jak to było w przypadku postaci, tylko że tym razem klikamy na ikonkę:



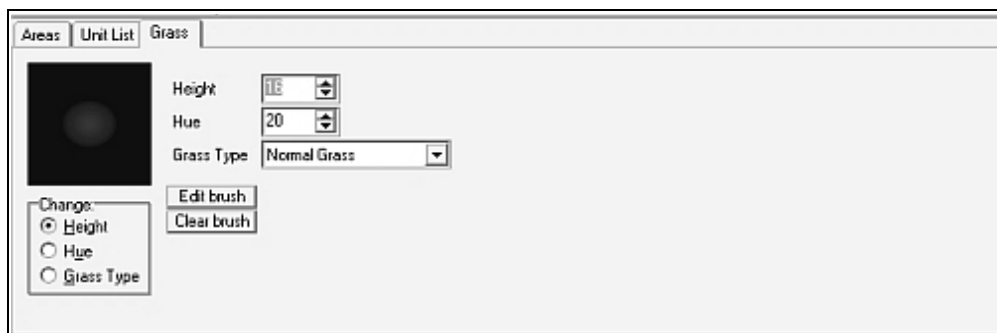
a następnie przytrzymać przycisk *SHIFT*, oraz kliknąć *PPM*, na obiekt, który chcemy usunąć.

### 3a. Dodawanie Trawy:

Aby dodać trawę do naszej mapy, wybieramy najpierw z paska narzędzi ikonkę:



następnie na dole okna programu pojawi się zakładka o nazwie "Grass", wyglądająca tak, jak na obrazku 6.3.



Rysunek 6.3 – Zakładka Grass

Są tu dostępne opcje:

- Change (są tu, do wyboru trzy opcje, Height (Wysoka) Hue (Zabarwienie), Grass Type (Typ Trawy), są to opcje podglądu typu trawy).
- Height (Wysokość, ustalamy tu wysokość trawy).
- Hue (Zabarwienie, ustalamy tu jaki to ma być odcień trawy).
- Grass Type (Typ Trawy, wybieramy tu jakiego gatunku to ma być trawa).
- Edit Brush (Edytuj Szczotkę, służy do edycji ustawień gęstości trawy).
- Clear Brush (Wyczyść Szczotkę, czyści ustawienia gęstości).

### 2b. Usuwanie Trawy:

Podobne, jak to było w przypadku postaci, oraz drzew, tylko klikamy inną ikonkę:



a następnie przytrzymać przycisk *SHIFT*, oraz kliknąć *PPM*, na obiekt, który chcemy usunąć.



## 4. Formowanie terenu:

Formowanie terenu, polega na podwyższaniu i obniżaniu go, trochę dokładniej, jest to opisane poniżej.

### **Podwyższanie:**

Aby podwyższyć teren najpierw wybieramy z paska narzędzi "Tools", ikonkę:



a następnie przytrzymujemy przycisk *SHIFT*, oraz klikamy w miejsca na mapie, które chcemy podnieść.

### **Obniżanie:**

Aby obniżyć teren najpierw wybieramy z paska narzędzi "Tools", ikonkę:



a następnie przytrzymujemy przycisk *SHIFT*, oraz klikamy w miejsca na mapie, które chcemy obniżyć.

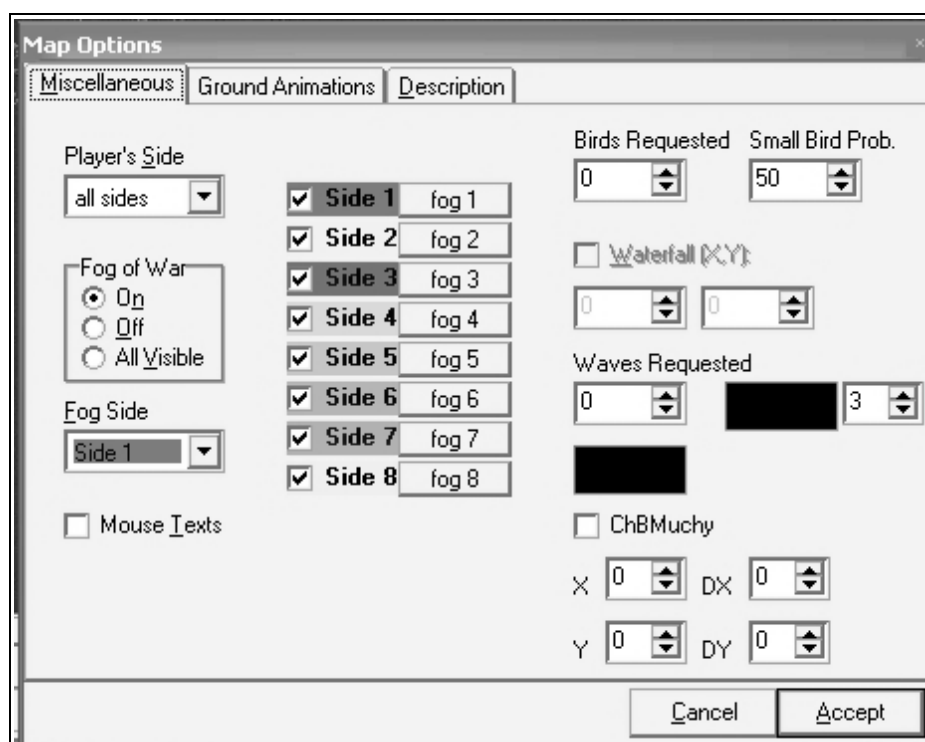
No i to by była cała filozofia (^\_^).

## 5. Ustawienia mapy, oraz Dobieranie Sojuszy:

### Ustawienia Mapy:

Są to ważne ustawienia, w których możemy wybrać np. jaki kolor będzie miała strona główna (czyli postać którą będziemy się poruszali, w czasie gry), kto jest z nami w sojuszu (czyli nam pomaga i nas nie atakuje), oraz kto jest dla nas wrogiem.

Ustawienia te uruchamiamy, za pomocą kliknięcia na klawiaturze przycisków **CTRL + J**, lub wybierając w menu **Files -> Map Preferences**. Po wybraniu któreś, z tych opcji, powinno wyświetlić się okno, takie jak na rysunku 6.5.1.



Rysunek 6.5.1 – Okno Map Options

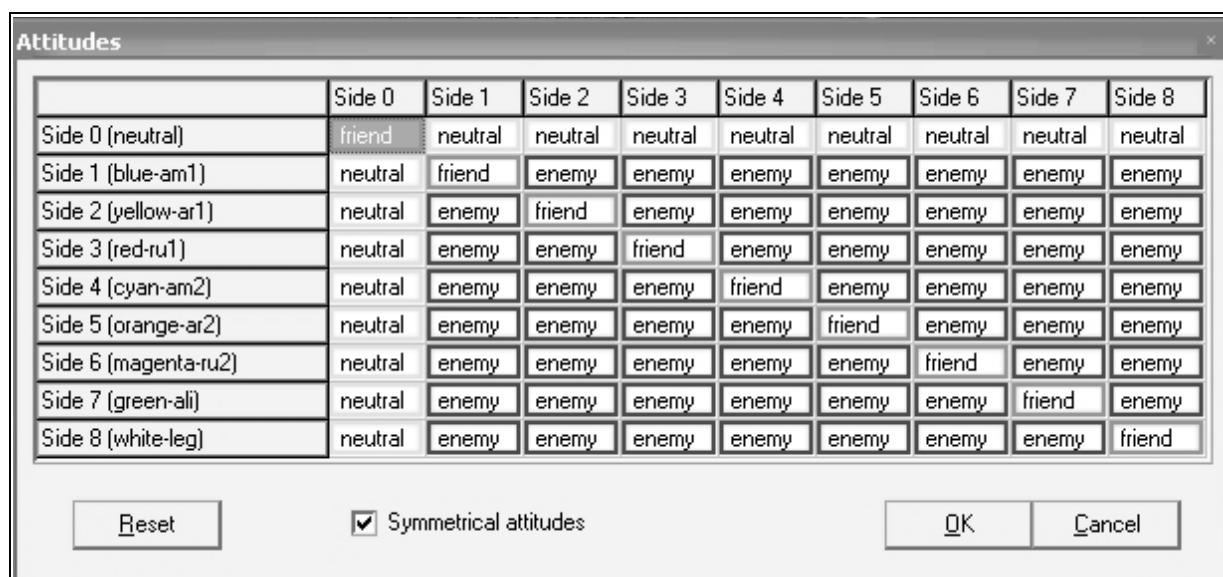
Są w nim dostępne opcje:

- Player's Side (Strona Gracza, tu wybieramy którą chcemy być strona, np. Side 1, Side 2, itd.).
- Fog of War (Mgła w Grze, są tu dostępne trzy możliwości: On (Włączona), Off (Wyłączona), All Visible (Wszystko Widoczne, ta opcja służy do wyboru, czy mgła ma być, czy jej nie ma być).
- Fog Side (Strona Mgły, służy do wyboru, który gracz ma widzieć tą samą mgłę, np. tak jak to jest w trybie multiplayer, gdy wybierzemy wspólny tryb widzenia).
- Mouse Texts (Teksty Myszy, zaznaczamy tu, czy po najechaniu na coś kursorem myszy ma się pojawić opis, czy nie).
- Panel kolorów stron (wybieramy w nim, która strona ma mieć jaki kolor).
- Oraz kilka opcji związanych, z zakładką "Description" (Opis).

Po skończonej pracy klikamy *Accept*.

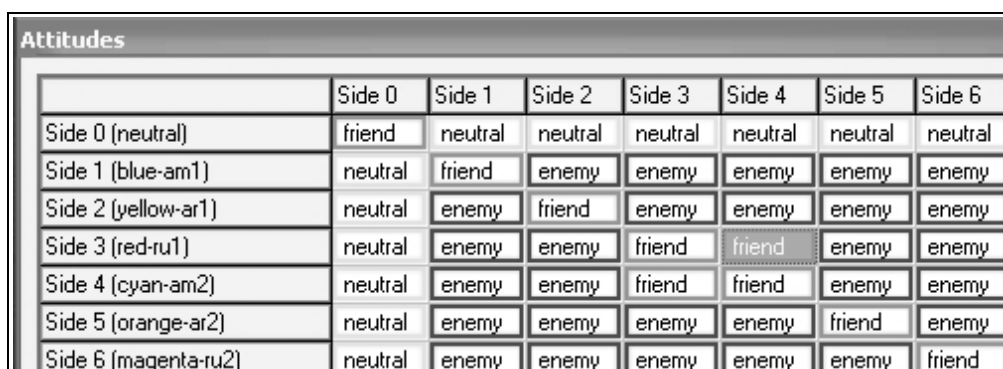
### Dobieranie Sojuszy:

Jeżeli już ustawiłeś, którą chcesz być stroną, oraz jaki chcesz mieć kolor, to teraz wybierz z menu **Mission -> Attitudes**. Powinno się pojawić takie okno jak na rysunku 6.5.2.



Rysunek 6.5.2 – Okno Attitudes

W tym oknie będziemy teraz ustawiać sojusze, aby to zrobić musimy poszukać w szeregu poziomych wierszy (np. Side 3 (red-ru1)) poszukać teamu, który chce mieć sojusz z kimś, oraz na szeregu pionowych kolumn (np. Side 4), poszukać teamu, z którym ten team chce mieć sojusz, oraz w miejscu przecinania się linii tych teamów, zmienić enemy, na friend (naciskając dwa razy na to miejsce), lub neutral. Przykładowy obrazek sojuszu, między side 3, oraz side 4, można zobaczyć na rysunku 6.5.3.



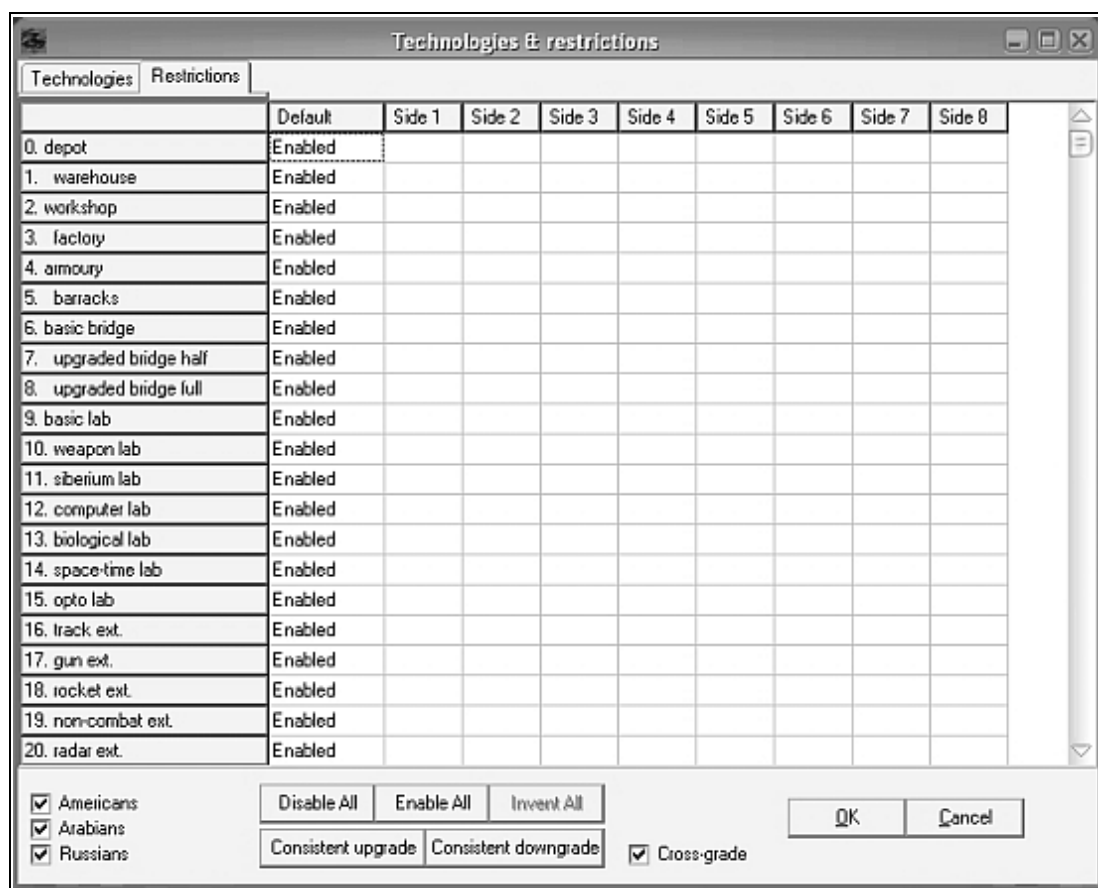
Rysunek 6.5.3 – Sojusz między teamami

Gdy skończymy, klikamy *OK*.

## 6. Dostępność budynków w misjach:

Dostępność budynków, to nic innego jak udostępnienie możliwości budowania, jakiegoś budynku w danej misji, czyli np. można wyłączyć możliwość budowania składu i wtedy nie będzie można w tej misji budować składu.

A więc do konkretów. Aby włączyć okno wyboru udostępnień, musimy wybrać w menu opcję: **Mission -> Restrictions**, wtedy powinno się pojawić okno, takie jak na rysunku 6.6.



Rysunek 6.6 – Okno wyboru udostępnień budowania

Aby zmienić opcję dostępności, musimy w tej tabeli wybrać przy danym budynku, opcję Enabled (Udostępnij), lub Disabled (Wyłączony), w zależności czy chcemy go udostępnić, czy nie.

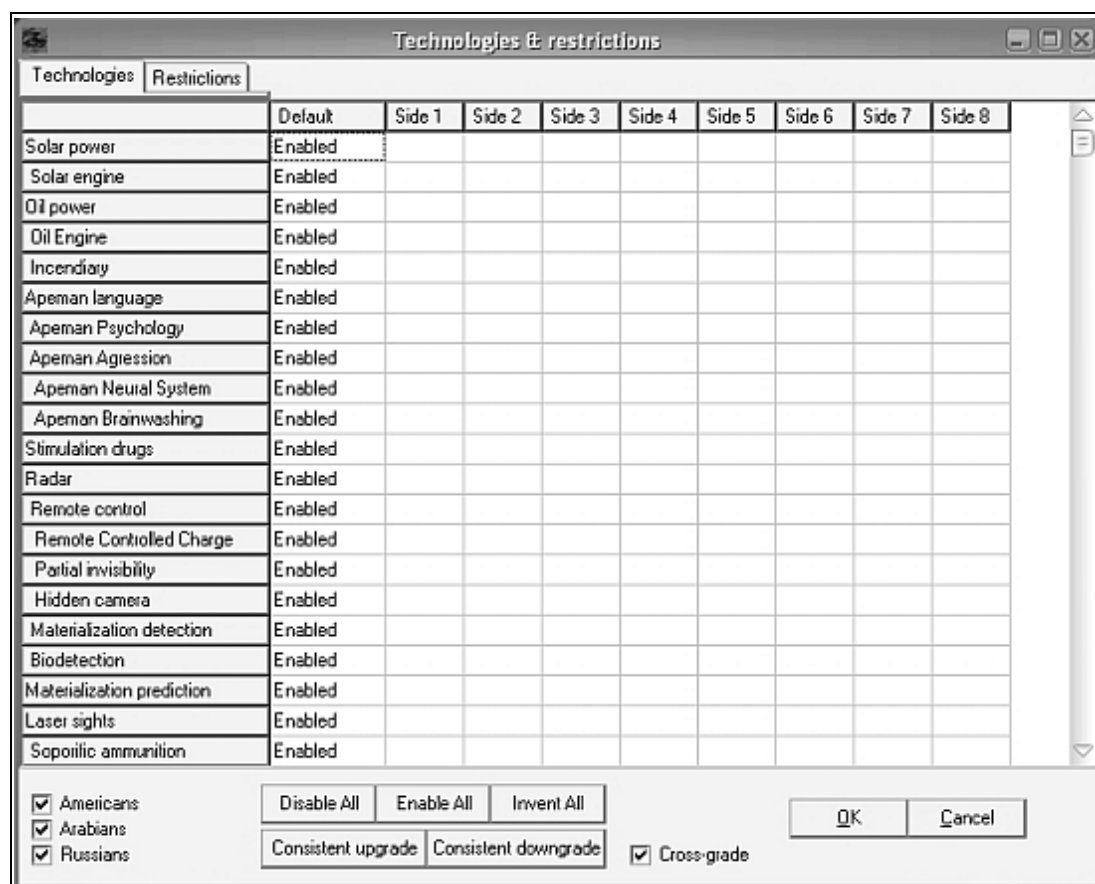
Są tu jeszcze przyciski dodatkowe, tj.:

- Disable All (Wyłącz Wszystkie)
- Enable All (Załącz Wszystkie)

Po skończonym zaznaczaniu, klikamy *OK*.

## 7. Dostępność technologii w misjach:

Z dostępnością technologii, jest podobnie jak z dostępnością budynków, różnica polega tylko na tym, że w menu wybieramy tylko jedną opcję niżej. A więc przejdźmy już, do szczegółów tej, że operacji. Najpierw wybieramy z menu programu opcję **Missions -> Technologies**, a następnie powinno się pojawić okienko widoczne, na rycinie 6.7.



Rysunek 6.7 – Okno wyboru udostępnień technologii

Podobnie, jak to było w poprzednim przykładzie, aby zmienić opcję dostępności, musimy w tej tabeli wybrać przy danej technologii, opcję Enabled (Udostępnić), lub Disabled (Wyłączyć), w zależności czy chcemy ją udostępnić, czy nie.

Są tu jeszcze przyciski dodatkowe, tj.:

- Disable All (Wyłącz Wszystkie)
- Enable All (Załącz Wszystkie)

Po skończonym zaznaczaniu, klikamy *OK*.

## \* **Skróty Klawiaturowe**

*Otworz* - Ctrl+O  
*Restart* - Ctrl+T  
*Zapisz* - Ctrl+S  
*Zapisz jako* - Shift+Ctrl+S  
*Zapisz kopie* - Ctrl+Alt+S  
*Ustawienia mapy* - Ctrl+J  
*Ustawienia edytora* - Shift+Ctrl+J  
*Eksport bitmapy podłoża* - Shift+Ctrl+R  
*Eksport bitmapy* - Shift+Ctrl+B  
*Eksport mapy wysokości wody* - Shift+Ctrl+W  
*Eksport wysokości mapy* - Shift+Ctrl+H  
*Import bitmapy* - Shift+Ctrl+M  
*Import wysokości wody mapy* - Shift+Ctrl+A  
*Import animacji wody mapy* - Shift+Ctrl+N

*Przejdź do trybu gry* - Ctrl+E

*Cofnij* - Alt+BkSp  
*Przywróć* - Shift+Alt+BkSp

*Zaznacz wszystko* - Ctrl+A  
*Odznacz* - Ctrl+D  
*Zaznacz przeciwne* - Shift+Ctrl+I  
*Ukryj zaznaczenie* - Ctrl+H  
*Rozszerz zaznaczenie* - Shift+Ctrl+E  
*Skurcz zaznaczenie* - Shift+Ctrl+C

*Warstwa Wysokości* - F1  
*Warstwa Terenu* - F2  
*Warstwa Trawy* - F3  
*Warstwa Środowiska* - F4  
*Warstwa Jednostki* - F5  
*Warstwa Info* - F6  
*Warstwa Wody* - F7  
*Warstwa wysokości widoczna* - Shift+F1  
*Warstwa terenu widoczna* - Shift+F2  
*Warstwa trawy widoczna* - Shift+F3  
*Warstwa środowiska widoczna* - Shift+F4  
*Warstwa jednostki widoczna* - Shift+F5  
*Warstwa info widoczna* - Shift+F6  
*Warstwa wody widoczna* - Shift+F7  
*Pokaz wszystkie warstwy* - Shift+F12

*Stworzyć nowy obszar* - Alt+Ins  
*Usunąć obszar* - Alt+Del  
*Zaznaczyć obszar* - End  
*Dodać obszar do zaznaczenia* - Shift+End  
*Podłożyć obszar z zaznaczenia* - Alt+End  
*Zestawić obszar z zaznaczenia* - Ctrl+End  
*Rozpoznać obszar* - Home  
*Dodać zaznaczenie do obszaru* - Shift+Home  
*Podłożyć zaznaczenie z obszaru* - Alt+Home  
*Zestawić zaznaczenie obszaru* - Ctrl+Home

*Zatwierdzić formowanie terenu* - Shift+Ctrl+T

*Ustaw punkty na* - Ctrl+Shift+Click

*Siatka widoczna* - Shift+Alt+G  
*Tryb sztucznego podłoża* - Shift+Alt+D  
*Rozdzielacz terenu* - Shift+Alt+R  
*Z-Buffer widoku* - Shift+Alt+Z  
*Przełącz FoW wszystkie widoczne* - Shift+Alt+F

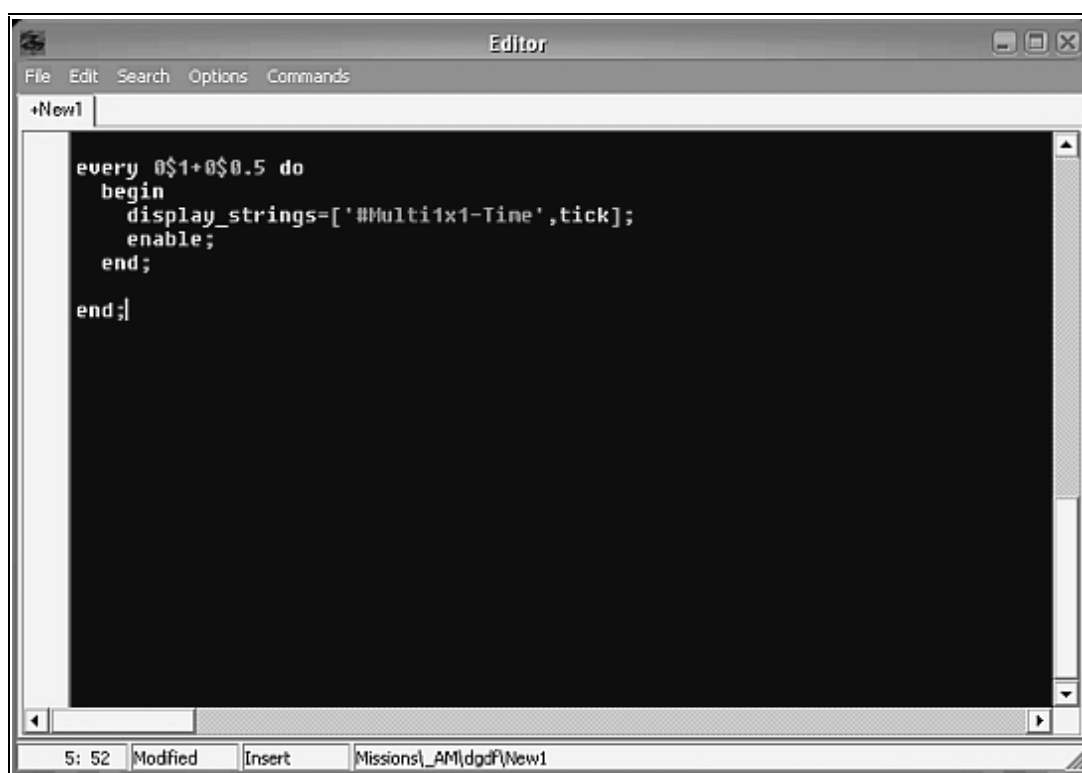
*Widoczny Sail Edytor* - Shift+Ctrl+'  
*Filtr trawy* - Shift+Ctrl+4  
*Widoczne ustawienia wewnętrzne* - Shift+Ctrl+8  
*Widoczne rozwijanie* - Shift+Ctrl+9  
*Widoczna minimapa* - Shift+Ctrl+-  
*Widoczne błędy* - Shift+Ctrl+D

## \* Sail, Jako Język Skryptowy

No i wreszcie ten długo oczekiwany rozdział, w którym to już zaczynają się schody z tworzeniem modów, do "Original War". W tym rozdziale zajmiemy się zagadnieniem *Sail Skryptu*, jego składnią, funkcjami, oraz stałymi.

### 1. Jak uruchomić Sail Editor?:

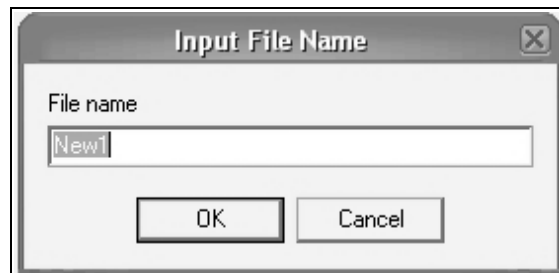
Aby włączyć edytor funkcji *Saila*, należy kliknąć w menu *Windows -> Sail Editor Visible*, a następnie pojawi się okno edytora *Sail Skryptu*. Sam edytor jest podobny w budowie do edytorów pascala, tj. Turbo Pascal, lub chociażby Free Pascal. Okno to jest widoczne na rysunku 8.1.1.



Rysunek 8.1.1 – Okno Sail Editora

Aby stworzyć dla mapy nowy dokument *Sail Skryptu*, musimy najpierw wybrać w *Sail Edytorze* z menu *File -> New Module*, po naciśnięciu tej opcji pojawi się okno z zapytaniem o nazwę modułu, czyli nazwę pliku, w którym będziemy zapisywali dane skrypty (standardowo jest to New1), okno widoczne na rysunku 8.1.2.





**Rysunek 8.1.2** – Okno Input File Name

W tym oknie wpisujemy nazwę pliku, a następnie klikamy przycisk "OK", jeżeli wszystko wykonaliśmy dobrze, to powinno się zmienić tło okna na niebieskie.

*Sail Editor*, ma w sobie kilka całkiem przydatnych funkcji, lecz o tym opowiem, w następnym punkcie tego rozdziału.

## 2. O Sail Edytorze słów kilka:

A więc, teraz opiszę trochę budowę okna *Sail Edytora*. A więc kolejno:

### **Menu Sail Edytora:**

Menu składa się z pięciu kategorii, File (Plik), Edit (Edycja), Search (Szukaj), Options (Opcje), Commands (Komendy), każda z nich odpowiada za coś innego, opiszę je teraz dokładnie:

#### **1. Menu Plik (File):**

- New Module – Nowy moduł
- Rename Module – Zmień nazwę modułu
- Save Module – Zapisz moduł
- Save All – Zapisz wszystko
- Remove Module – Usuń moduł

#### **2. Menu Edycja (Edit):**

- Undo – Cofnij
- Cut – Wytnij
- Copy – Kopiuj
- Paste – Wklej
- Delete – Kasuj
- Select All – Zaznacz wszystko

#### **3. Menu Szukaj (Search):**

- Find – Znajdź
- Replace – Zmień
- Find Next – Znajdź następne

#### **4. Menu Opcje (Options):**

- Persistent Block - Kursor dopisujący
- Overwrite Block - Kursor nadpisujący

#### **5. Menu Komendy (Commands):**

- Run – Uruchom
- Compile – Kompiluj
- Trace – Śledź
- Program Reset – Zresetuj program
- Focus Hex – Skupienie Hex

### **Pasek stanu:**

Jest podzielony na cztery części, można na nim sprawdzić, np. w której linii kodu jesteśmy, albo w jakim katalogu jest zapisany dany moduł.

### **Kolorowanie składni:**

Program zawiera kolorowanie składni, czyli podkreślanie składni kilkoma kolorami, itd., co może bardzo ułatwić pisanie kodu.

### 3. Składnia Saila:

Najważniejszym punktem w każdym języku jest składnia. Składnia, to nic innego jak słowa kluczowe, czyli elementy jakiegoś języka sterujące pracą programu, oraz znaki, które są równie ważne jak i same słowa kluczowe i trzeba je zapisywać w wyżej określonym porządku.

Prezentowana przez nas składnia języka skryptowego *Sail*, jest podobna do tej znanej z języka Delphi, gdyż właśnie *Sail* powstał na podstawie Delphi.

#### **Podstawy składni:**

Kod źródłowy złożony jest z poleceń zakończonych wyżej określonymi znakami. *Sail*, tak jak i każdy inny język ma za zasadę to, iż nie można w nim bałaganić, gdyż każdy błąd w pisowni, lub zapomnienie jakiegoś znacznika najczęściej na końcu polecenia, może się skończyć tym, iż nasz mod będzie jedynie wyświetlał błąd i nic więcej. Dlatego trzeba pamiętać, iż kod to nie śmietnik i musi tam być porządek.

#### **Wielkie i małe litery w Sailu?:**

W *Sailu*, podobnie jak w Delphi nie jest ważnym to, czy używasz dużych, czy małych liter do pisania kodu (i to jest jeden z jego plusów). Więc polecenia tj. *InGameOff* możemy napisać w wiele różnych sposobów, oto przykłady:

ingameoff ingameoff INGAMEOFF InGaMeOfF itd...
--

Jednak nie jest zbyt zalecanym sposobem pisanie kodu, gdy używamy jedynie małych liter, nie stosujemy wcięć w kodzie, gdyż wtedy kod zaczyna wyglądać jak jeden wielki śmietnik, w którym nikt się nie odnajdzie po wpisaniu paruset, lub paru tysięcy linijek kodu.

#### **Średnik? A co to takiego?...:**

Jak już pisałem wcześniej "Kod źródłowy złożony jest z poleceń zakończonych wyżej określonymi znakami", a więc powiem coś o najważniejszym z nich. *Sail* tak samo jak i Delphi, mają jedną żelazną zasadę, która brzmi: "Każda instrukcja musi być zakończona znakiem średnika (;)." i tego trzeba się trzymać, gdyż zapomnienie gdziekolwiek w kodzie jakiegoś średnika, może spowodować, to iż nasz mod w ogóle nie ruszy...

#### **Co to są bloki begin i end? I do czego służą?:**

Każdy dobrze napisany kod programu jest zawsze wpisany pomiędzy bloki *begin* i *end*, gdzie *end* jest zawsze zakończone średnikiem (;).

Program w czasie wczytywania mapy wczytuje zawsze instrukcje rozpoczęte słowem *begin*, czyli tak jakby początek kodu, który ma zostać wykonany, oraz kończy się na słowie *end*.

Należy, też pamiętać, iż liczba instrukcji *begin* nie może być większa od liczby instrukcji *end*, oraz

na odwrót. Gdyż wtedy wystąpi błąd w grze.

Oto przykład, jak powinna wyglądać budowa kodu poprawnego:

```
begin  
begin  
  
end;  
end;
```

## 4. Stałe Saila:

Podobnie do zmiennych, tak i *stałe* służą do przechowywania danych w czasie działania aplikacji. Jednak jest między nimi dosyć duża różnica, jak sama nazwa (*STAŁE*) wskazuje, iż nie mogą one podlegać zmodyfikowaniu w czasie działania programu, a ich wartość jest z góry określona. W *Sailu* mamy te udogodnienie, iż w większości przypadków, jeżeli używamy stałych, to nie musimy pisać ich wartości, gdyż gra sama wie jakie ma podać.

### A to lista stałych w *Sailu*:

```
unit_human = 1
unit_vehicle = 2
unit_building = 3
unit_crate = 4
nation_nature = 0
nation_american = 1
nation_arabian = 2
nation_russian = 3
nation_arabian_music = 4
hranice_umirani = 250
hranice_zraneni = 500
hranice_zdravi = 1000
sex_male = 1
sex_female = 2
attr_stamina = 1
attr_speed = 2
skill_combat = 1
skill_engineering = 2
skill_mechanical = 3
skill_scientistic = 4
class_soldier = 1
class_engineer = 2
class_mechanic = 3
class_scientistic = 4
class_sniper = 5
class_mortar = 8
class_bazooker = 9
class_desert_warrior = 11
class_apeman = 12
class_baggie = 13
class_tiger = 14
class_apeman_soldier = 15
class_apeman_engineer = 16
class_apeman_kamikaze = 17
class_phororhacos = 18
class_frog = 19
class_fish = 20
```

*classtype\_soldier* = 1  
*classtype\_engineer* = 2  
*classtype\_mechanic* = 3  
*classtype\_scientist* = 4  
*classtype\_noble* = 5  
*classtype\_apeman* = 6  
*classtype\_critter* = 7  
*classtype\_tiger* = 8  
*b\_depot* = 0  
*b\_warehouse* = 1  
*b\_workshop* = 2  
*b\_factory* = 3  
*b\_armoury* = 4  
*b\_barracks* = 5  
*b\_lab* = 6  
*b\_lab\_half* = 7  
*b\_lab\_full* = 8  
*b\_lab\_basic* = 9  
*b\_lab\_weapon* = 10  
*b\_lab\_siberium* = 11  
*b\_lab\_computer* = 12  
*b\_lab\_biological* = 13  
*b\_lab\_spacetime* = 14  
*b\_lab\_opto* = 15  
*b\_ext\_track* = 16  
*b\_ext\_gun* = 17  
*b\_ext\_rocket* = 18  
*b\_ext\_noncombat* = 19  
*b\_ext\_radar* = 20  
*b\_ext\_siberium* = 21  
*b\_ext\_radio* = 22  
*b\_ext\_stitch* = 23  
*b\_ext\_computer* = 24  
*b\_ext\_laser* = 25  
*b\_oil\_power* = 26  
*b\_solar\_power* = 27  
*b\_siberite\_power* = 28  
*b\_oil\_mine* = 29  
*b\_siberite\_mine* = 30  
*b\_breastwork* = 31  
*b\_bunker* = 32  
*b\_turret* = 33  
*b\_teleport* = 34  
*b\_fort* = 35  
*b\_control\_tower* = 36  
*b\_eon* = 38  
*b\_behemoth* = 37

*b\_alien\_tower* = 39  
*engine\_combustion* = 1  
*engine\_solar* = 2  
*engine\_siberite* = 3  
*control\_manual* = 1  
*control\_remote* = 2  
*control\_computer* = 3  
*control\_rider* = 4  
*control\_apeman* = 5  
*us\_light\_wheeled* = 1  
*us\_medium\_wheeled* = 2  
*us\_medium\_tracked* = 3  
*us\_heavy\_tracked* = 4  
*us\_morphling* = 5  
*us\_machine\_gun* = 2  
*us\_light\_gun* = 3  
*us\_gatling\_gun* = 4  
*us\_double\_gun* = 5  
*us\_heavy\_gun* = 6  
*us\_rocket\_launcher* = 7  
*us\_siberium\_rocket* = 8  
*us\_siberium\_rocket\_remainder* = 15  
*us\_laser* = 9  
*us\_double\_laser* = 10  
*us\_radar* = 11  
*us\_cargo\_bay* = 12  
*us\_crane* = 13  
*us\_bulldozer* = 14  
*ar\_hovercraft* = 11  
*ar\_light\_trike* = 12  
*ar\_medium\_trike* = 13  
*ar\_half\_tracked* = 14  
*ar\_multimissile\_ballista* = 22  
*ar\_light\_gun* = 23  
*ar\_double\_machine\_gun* = 24  
*ar\_gatling\_gun* = 25  
*ar\_flame\_thrower* = 26  
*ar\_gun* = 27  
*ar\_rocket\_launcher* = 28  
*ar\_selfpropelled\_bomb* = 29  
*ar\_radar* = 30  
*ar\_control\_tower* = 31  
*ar\_cargo\_bay* = 32  
*ru\_medium\_wheeled* = 21  
*ru\_medium\_tracked* = 22  
*ru\_heavy\_wheeled* = 23  
*ru\_heavy\_tracked* = 24

*ru\_heavy\_machine\_gun* = 42  
*ru\_gatling\_gun* = 43  
*ru\_gun* = 44  
*ru\_rocket\_launcher* = 45  
*ru\_heavy\_gun* = 46  
*ru\_rocket* = 47  
*ru\_siberium\_rocket* = 48  
*ru\_siberium\_rocket\_remainder* = 55  
*ru\_time\_lapser* = 49  
*ru\_cargo\_bay* = 51  
*ru\_crane* = 52  
*ru\_bulldozer* = 53  
*att\_neutral* = 0  
*att\_friend* = 1  
*att\_enemy* = 2  
*mat\_cans* = 1  
*mat\_oil* = 2  
*mat\_siberit* = 3  
*mat\_none* = 0  
*mat\_artefact* = 4  
*mat\_artifact* = 4  
*mat\_multi* = 5  
*source\_oil* = 1  
*source\_siberium* = 2  
*cargo\_unit* = 10  
*tech\_SolPow* = 35  
*tech\_SolEng* = 45  
*tech\_OilPow* = 46  
*tech\_OilEng* = 47  
*tech\_ApeLang* = 1  
*tech\_ApePsych* = 2  
*tech\_ApeAgres* = 11  
*tech\_ApeNeural* = 3  
*tech\_ApeBrain* = 4  
*tech\_StimDrugs* = 5  
*tech\_Radar* = 6  
*tech\_MatDet* = 7  
*tech\_BioDet* = 8  
*tech\_MatPred* = 9  
*tech\_LasSight* = 12  
*tech\_Soporific* = 13  
*tech\_Laser* = 10  
*tech\_LasDouble* = 14  
*tech\_RemCont* = 15  
*tech\_RemCharge* = 18  
*tech\_PartInvis* = 16  
*tech\_HidCam* = 17



*tech\_SibDet* = 20  
*tech\_SibLoc* = 19  
*tech\_SibPow* = 21  
*tech\_SibEng* = 22  
*tech\_Behemoth* = 23  
*tech\_Artifact* = 24  
*tech\_SibFiss* = 25  
*tech\_SibContam* = 26  
*tech\_TauRad* = 28  
*tech\_SpacAnom* = 29  
*tech\_TauField* = 30  
*tech\_Lapser* = 31  
*tech\_LimTeleport* = 37  
*tech\_TargTeleport* = 38  
*tech\_AI* = 32  
*tech\_AdvAI* = 27  
*tech\_Virus* = 33  
*tech\_AdvChassis* = 36  
*tech\_Flame* = 70  
*tech\_Gatling* = 69  
*tech\_Gun* = 39  
*tech\_AdvMet* = 34  
*tech\_Rocket* = 40  
*tech\_AdvRocket* = 71  
*tech\_Mortar* = 41  
*tech\_Explos* = 42  
*tech\_SelfDest* = 43  
*tech\_Bazooka* = 44  
*tech\_Tech1* = 48  
*tech\_Tech2* = 49  
*tech\_Tech3* = 50  
*tech\_Weap1* = 51  
*tech\_Weap2* = 52  
*tech\_Weap3* = 53  
*tech\_Sib1* = 54  
*tech\_Sib2* = 55  
*tech\_Sib3* = 56  
*tech\_Comp1* = 57  
*tech\_Comp2* = 58  
*tech\_Comp3* = 59  
*tech\_Opto1* = 60  
*tech\_Opto2* = 61  
*tech\_Opto3* = 62  
*tech\_ST1* = 63  
*tech\_ST2* = 64  
*tech\_ST3* = 65  
*tech\_Bio1* = 66

*tech\_Bio2* = 67  
*tech\_Bio3* = 68  
*state\_disabled* = 0  
*state\_enabled* = 1  
*state\_researched* = 2  
*f\_and* = 1  
*f\_or* = 2  
*f\_not* = 3  
*f\_type* = 21  
*f\_side* = 22  
*f\_nation* = 23  
*f\_lives* = 24  
*f\_class* = 25  
*f\_sex* = 26  
*f\_minskill* = 28  
*f\_maxskill* = 29  
*f\_btype* = 30  
*f\_chassis* = 31  
*f\_engine* = 32  
*f\_control* = 33  
*f\_weapon* = 34  
*f\_bweapon* = 35  
*f\_ok* = 50  
*f\_alive* = 51  
*f\_placed* = 52  
*f\_ready* = 53  
*f\_inside* = 54  
*f\_driving* = 55  
*f\_outside* = 56  
*f\_constructed* = 57  
*f\_empty* = 58  
*f\_occupied* = 59  
*f\_hastask* = 60  
*f\_linked* = 61  
*f\_enemy* = 81  
*f\_ally* = 82  
*f\_neutral* = 83  
*f\_dist* = 91  
*f\_distxy* = 92  
*f\_inarea* = 95  
*f\_exceptarea* = 96  
*f\_see* = 101  
*bfo\_defend\_bonus\_human* = 1  
*bfo\_defend\_bonus\_vehicle* = 2  
*bfo\_defend\_bonus\_building* = 3  
*bfo\_range* = 10  
*bfo\_height* = 20

*sel\_hired* = -1  
*sel\_not\_hired* = -2  
*sel\_changeable* = -3  
*sel\_not\_changeable* = -4  
*sel\_change\_class* = -5  
*sel\_dont\_change\_class* = -6  
*sel\_ignore\_class\_nation* = -7  
*sel\_dont\_ignore\_class\_nation* = -8  
*mc\_move\_wait* = 1  
*mc\_move\_through* = 2  
*mc\_move\_waitwp* = 3  
*mc\_move\_dontshoot* = 4  
*mc\_move\_dontcapture* = 5  
*mc\_move\_agressive* = 6  
*mc\_move\_ownstuck* = 7  
*mc\_base\_cont* = 1  
*mc\_reg\_refresh\_time* = 1  
*mc\_reg\_area\_to\_guard* = 2  
*mc\_reg\_area\_to\_protect* = 3  
*mc\_reg\_units\_to\_protect* = 4  
*mc\_reg\_units\_to\_guard* = 10  
*mc\_reg\_expire\_stops\_to\_attack* = 5  
*mc\_reg\_expire\_leaves\_area* = 6  
*mc\_reg\_ignore\_fog* = 7  
*mc\_reg\_only\_important* = 8  
*mc\_reg\_buildings* = 9  
*mc\_def\_advantage* = 1  
*mc\_area\_dont\_leave* = 2  
*mc\_def\_change\_to\_soldiers* = 3  
*mc\_def\_change\_to\_mechanics* = 4  
*mc\_retreat\_lives\_people* = 5  
*mc\_retreat\_lives\_vehicles* = 6  
*mc\_retreat\_area\_people* = 7  
*mc\_retreat\_area\_vehicles* = 8  
*mc\_out\_of\_fuel* = 9  
*mc\_no\_stop* = 10  
*mc\_pat\_aggressive* = 11  
*mc\_murder* = 12  
*mc\_done\_killed\_by\_reg* = -2  
*mc\_done\_killed* = -1  
*mc\_done\_normal* = 0  
*mc\_done\_no\_people* = 1  
*mcs\_Build* = 1  
*mcs\_Upgrade* = 2  
*mcs\_Tech* = 3  
*mcs\_Veh* = 4  
*mcs\_Fort* = 5

*mcs\_Fort\_Upgrade* = 6  
*mcs\_Mat* = 7  
*mcs\_Blank* = 8  
*mcs\_depot\_pos* = 1  
*mcs\_armoury\_pos* = 2  
*mcs\_fact\_pos* = 3  
*mcs\_lab\_pos* = 4  
*mcs\_power\_pos* = 5  
*mcs\_defend\_pos* = 6  
*mcs\_ext\_pos* = 7  
*mcs\_oilmine\_pos* = 8  
*mcs\_sibmine\_pos* = 9  
*mcs\_teleport\_pos* = 10  
*mcs\_tower\_pos* = 11  
*mcs\_behemoth\_pos* = 12  
*mcs\_PosList* = 1  
*mcs\_Strategy* = 2  
*mcs\_Parking* = 3  
*mcr\_hum\_area* = 1  
*mcr\_veh\_area* = 2  
*mcr\_area\_only* = 3  
*mcr\_hum\_limit* = 4  
*mcr\_veh\_limit* = 5  
*mcr\_bui\_limit* = 6  
*mcr\_safety* = 7  
*mcr\_repv\_area* = 8  
*mcr\_capv\_area* = 9  
*mcr\_repb\_area* = 10  
*mcr\_capb\_area* = 11  
*mcw\_hum\_lo* = 4  
*mcw\_hum\_hi* = 5  
*mcw\_veh\_lo* = 6  
*mcw\_veh\_hi* = 7  
*mcw\_hum\_area* = 1  
*mcw\_veh\_area* = 2  
*mcw\_corpse\_area* = 8  
*bs\_error* = 0  
*bs\_build* = 1  
*bs\_idle* = 2  
*bs\_working* = 3  
*bs\_pause* = 4  
*bs\_suspend* = 5  
*bs\_need\_people* = 6  
*bs\_need\_power* = 7  
*bs\_need\_extension* = 8  
*bs\_need\_ape* = 10  
*bs\_waiting* = 9

*art\_no* = 0  
*art\_gray* = 1  
*art\_instant* = 2  
*art\_place* = 3  
*art\_unit* = 4  
*art\_human* = 5  
*art\_vehicle* = 6  
*art\_building* = 7  
*art\_friend\_unit* = 8  
*art\_friend\_human* = 9  
*art\_friend\_vehicle* = 10  
*art\_friend\_building* = 11  
*art\_enemy\_unit* = 12  
*art\_enemy\_human* = 13  
*art\_enemy\_vehicle* = 14  
*art\_enemy\_building* = 15  
*art\_exp\_left* = 1  
*art\_exp\_mid* = 2  
*art\_exp\_right* = 3  
*art\_use\_eye* = 4  
*art\_use\_sibexplosion* = 5  
*art\_use\_teleport* = 6  
*art\_use\_atom* = 7  
*art\_use\_cube* = 8  
*art\_use\_power* = 9  
*art\_use\_human* = 10  
*art\_use\_dead* = 11  
*art\_use\_exclamation* = 12  
*art\_use\_hand* = 13  
*art\_use\_tau* = 14  
*art\_use\_sibdestruct* = 15  
*music\_auto* = 0  
*music\_combat* = 1  
*music\_prep* = 2  
*music\_recon* = 3  
*music\_victory* = 4  
*music\_menu* = 5

## 5. Funkcje Saila:

Funkcje to nic innego, jak komendy, które musimy wpisać, aby gra po wczytaniu ich dopuściła do jakiejś akcji, np. pojawienie się małpoluda, budowanie się bazy po stronie komputera, lub pojawienie się dialogu.

Aktualnie jest to oryginalny spis funkcji udostępniony przez firmę deweloperską *Altar Interactive* (twórcę gry), więc jest to aktualnie spis w języku angielskim, ale w przyszłości, jak ktoś to spolszczy, lub jak ja będę miał czas i to spolszczę, to ukaże się tutaj polska wersja tego spisu funkcji, ale i tak powinniście sobie poradzić z zrozumieniem ich, gdyż są pisane prosto.

### Oto spis Funkcji Saila:

#### **Replace (list:plist; index:integer; item:tvalue)**

replaces item of list with specified index with another, returns result

if  $\text{index} > n$  ( $n = \text{length of list}$ ), then positions  $n+1$  to  $\text{index}-1$  are filled by undef value

#### **Insert(list:plist index:integer item:tvalue)**

inserts item to list at specified position (1 - before first, n - before n-th), returns result

if  $\text{index} > n+1$  ( $n = \text{length of list}$ ), then positions  $n+2$  to  $\text{index}-1$  are filled by undef value

#### **Delete(list:plist index:integer)**

deletes item from list, returns result

if index is out of list, then happens nothing

#### **ExclusiveOn**

stops all other threads of SAIL until ExclusiveOff is executed or this thread ends

#### **ExclusiveOff**

cancels effect of ExclusiveOn

#### **DialogueOn**

function, which: store tick, switch stop\_action to true, set ExclusiveOn

#### **DialogueOff**

function, which: restore tick, switch stop\_action to false, set ExclusiveOff

#### **InGameOn**

function, which: store tick, switch ingame\_video to true, set ExclusiveOn

### **InGameOff**

function, which: restore tick, switch ingame\_video to false, set ExclusiveOff

### **Randomize**

randomize SAIL random numbers

### **RandomizeAll**

randomize all random numbers in game

### **Rand(min:integer max:integer)**

random number in selected range returned

### **Prob(per:integer)**

returns true on per percent ( $\text{Rand}(1,100) \leq \text{per}$ )

### **MultiRand(min:integer max:integer n:integer)**

random number in selected range, made by n Rand

### **RandHex(border:boolean)**

returns coordinate of random hex anywhere on map in format [x,y]

if border is true, then bordering hexes can be included

### **RandHexArea(area:integer border:boolean)**

returns coordinate of random hex in specified area in format [x,y] (or 0 if fails)

if border is true, then bordering hexes can be included

### **RandHexXYRx:integer y:integer r:integer border:boolean)**

returns coordinate of random not invalid hex near to xy (max. dist is R) in format [x,y] (or 0 if fails)

if border is true, then bordering hexes can be included

### **InitUc**

Initializes unit create structure to default values

### **InitHc**

Initializes human create structure to default values

### **InitVc**

Initializes vehicle create structure to default values

### **InitBc**

Initializes building create structure to default values

### **SaveForQuickRestart**

use it after ingame videos and character selection - quick restart will start from this point

### **NewCharacter(ident:string)**

Loads and create a new global character (which was never saved before), returns its handle

### **PrepareNewCharacter(ident:string)**

Prepares hc\_ and uc\_ structures for a new global character (which was never saved before)

### **TestVariable(ident:string)**

Returns true if there is a variable with specified ident in the save

### **TestCharacters(ident:string)**

Returns true if there is a character or group of characters with specified ident in the save

### **CheckCharacterSet(ident:string)**

Returns number of characters in character set from the save

(if there is an empty set in the save, CheckCharacterSet returns 0, but TestCharacter returns true)

### **LoadVariable(ident:string default:tvalue)**

Loads a variable with specified ident from the save (or default if ident is not found)

### **CreateCharacterSet(ident:string)**

Loads a set of characters with specified ident from the save

Creates people from it (except dead characters) using some uc\_side and returns set of its handles

### **CreateOneCharacterFromSet(ident:string index:integer)**

Same as CreateCharacterSet, but only one unit is created (the one with specified index in the set)

### **CreateOneCharacterFromSetWithClass(ident:string index:integer cl:integer)**

Same as CreatePeopleFromCharacterSet, but character gets specified class



**CreateCharacter(ident:string)**

Shortcut for CreateOneCharacterFromSet(ident,1)

**CreateCharacterWithClass(ident:string cl:integer)**

Shortcut for CreateOneCharacterFromSetWithClass(ident,1,cl)

**PrepareOneCharacterFromSet(ident:string index:integer)**

Same as CreateOneCharacter, but character is only prepared (not created yet), returns true if succeeds

**PrepareCharacter(ident:string)**

Shortcut for PrepareOneCharacterFromSet(ident,1)

**SaveCharacters(humlist:plist ident:string)**

Saves set of human characters to the output save and gives it specified ident

For special characters use:

SaveCharacters(handle,name) in one mission and handle:=CreateCharacter(name) in the next mission

For survived people from previous mission use

SaveCharacters(set,name) in one mission and set:=CreateCharacterSet(name) in the next mission

**SaveVariable(v:tvalue ident:string)**

Saves variable to the save with specified ident

**DeleteCharacters(ident:string)**

Deletes character set from the save

**DeleteVariable(ident:string)**

Deletes a character from the save

**CharacterSelection(ident:string min:integer max:integer units:plist classes:plist)**

Displays Character selection menu and returns list of characters selected by player

- ident is identifier of caption from Texts.txt

- min and max are minimum and maximum number of hired character to close dialog

- units is list of units, which can contain special commands which affects all following units:

    sel\_hired - following character are hired at start (default)

    sel\_not\_hired - following character are not hired at start

sel\_changeable - player can fire/hire following characters

sel\_not\_changeable - player cannot fire/hire following characters (default)

sel\_change\_class - player can change class of following characters

sel\_dont\_change\_class - player cannot change class of following characters (default)

sel\_ignore\_class\_nation - following characters can be changed to class independent of its nation

sel\_dont\_ignore\_class\_nation - following characters can be changed only to classes of its nation (default)

- classes is list of possibly classes to which can be units changed. Every item may be in two formats:

class\_number - this class can be used without limitations

[class\_number,min] - you must select at least min characters with this class

[class\_number,min,max] - you must select min to max characters with this class

Example1 (select up to five mechanic and soldiers from gamma2 and gamma3):

```
CharacterSelection('S1',1,6,  
    [JMM, sel_not_hired, sel_changeable] ^ gamma2 ^ gamma3,  
    [class_soldier, class_mechanic])
```

Example2 (select up to five men from gamma2 to help John and Tim build a base)

```
CharacterSelection('S2',3,7,  
    [sel_changeable, JMM, sel_not_changeable, Tim, sel_not_hired, sel_changeable] ^ gamma2,  
    [class_soldier, class_mechanic, [class_engineer,1], class_scientific])
```

### **RewardPeople(humans:plist)**

gives mission reward to list of human characters (use before save)

### **CreateHuman**

Creates human using uc\_ and hc\_ parameters

### **CreateVehicle**

Creates vehicle using uc\_ and vc\_ parameters

### **CreateBuilding**

Creates a building using uc\_ and bc\_ parameters

### **CreateAndPlaceBuildingXYD(x:integer y:integer direction:integer)**

Creates a building using uc\_ and bc\_ parameters

### **PlaceUnitXY(un:integer x:integer y:integer materialisation:boolean)**

Places unit at specified coordinates, returns true if succeeds

If materialisation is true, unit is materialised

**PlaceUnitArea(un:integer area:integer materialisation:boolean)**

Places unit into specified area, returns true if succeeds

If materialisation is true, unit is materialised

**PlaceUnitXYR(un:integer x:integer y:integer r:integer materialisation:boolean)**

Places unit near specified coordinates (max. dist is r), returns true if succeeds

If materialisation is true, unit is materialised

**PlaceUnitAnywhere(un:integer materialisation:boolean)**

Places unit anywhere on map, returns true if succeeds

If materialisation is true, unit is materialised

**PlaceHumanInUnit(hum:integer un:integer)**

Places unplaced human into a placed or unplaced unit

**LinkVehicleToHuman(veh:integer hum:integer)**

Links vehicle to a human in Control tower

**CreateCratesXY(amount:integer x:integer y:integer materialisation:boolean)**

No Description.

**CreateCratesArea(amount:integer area:integer materialisation:boolean)**

No Description.

**CreateCratesXYR(amount:integer x:integer y:integer r:integer materialisation:boolean)**

No Description.

**CreateCratesAnywhere(amount:integer materialisation:boolean)**

No Description.

**CreateResourcesXY(typ:integer amount:integer x:integer y:integer materialisation:boolean)**

No Description.

**CreateResourcesArea(typ:integer amount:integer area:integer materialisation:boolean)**

No Description.

**CreateResourcesXYR(typ:integer amount:integer x:integer y:integer r:integer materialisation:boolean)**

No Description.

**CreateResourcesAnywhere(typ:integer amount:integer materialisation:boolean)**

No Description.

**CreateDepositXY(x:integer y:integer typ:integer)**

creates deposit of mat\_siberite or mat\_oil type on specified coordinates

**RemoveDepositXY(x:integer y:integer)**

removes deposit of any type on specified coordinates

**RemoveUnit(un:integer)**

Removes specified unit from map

**DestroyUnit(un:integer)**

Destroys specified unit (include removing, if placed)

**KillUnit(un:integer)**

Kills specified unit by setting its lives to zero

**Wait(ticks:integer)**

waits for specified number of ticks, ignores asynchronous mode

**DWait(ticks:integer)**

waits for specified number of ticks, for use when DialogOn

**FilterAllUnits(filter:plist)**

filters list of all units using filter

**FilterUnitsInArea(area:integer filter:plist)**

filters list of units in specified area using filter

**FilterUnitsExceptArea(area:integer filter:plist)**

filters list of units not in specified area using filter

**UnitFilter(units:plist filter:plist)**

filters list of unit using filter

**NearestUnitToXY(units:plist x:integer y:integer)**

Returns nearest unit from list units to position (x,y), or zero if units is empty

**NearestUnitToUnit(units:plist un:integer)**

Returns nearest unit from list units to unit un (ignoring un itself), or zero

**AllNearestUnitToUnit(units:plist un:integer)**

Returns all nearest unit from list units to unit un (ignoring un itself), or zero

**SortListByListAsc(items:tvalue values:tvalue)**

Returns list items ascending sorted by asociated integer list values

If lists aren't the same size, only smaller length of list is used

**SortListByListDesc(items:tvalue values:tvalue)**

Returns list items descending sorted by asociated integer list values.

If lists aren't the same size, only smaller length of list is used

**WorstFromListByList(items:tvalue values:tvalue)**

Returns SortListByListAsc(items,values)[1], or zero iv values is empty. Faster than SortListByListDesc

**BestFromListByList(items:tvalue values:tvalue)**

Returns SortListByListDesc(items,values)[1], or zero iv values is empty. Faster than SortListByListAsc

**SetAttitude(fromside:integer toside:integer att:integer sym:boolean)**

sets attitude to att (if sym then sets the symetric relation too)

**GetAttitude(fromside:integer toside:integer)**

returns attitude

**SetAlliedVictory(side:integer state:boolean)**

sets allied victory state for side

**GetAlliedVictory(side:integer)**

returns allied victory state of side

**CenterOnXY(x:integer y:integer)**

centers screen on specified coordinates with scrolling

**CenterOnUnits(units:plist)**

centers screen to specified group of units with scrolling

**CenterNowOnXY(x:integer y:integer)**

centers immediately screen on specified coordinates

**CenterNowOnUnits(units:plist)**

centers immediately screen to specified group of units

**Say(un:integer ident:string)**

Specified unit says sentence identified by ident

**SayNoFace(un:integer ident:string)**

Specified unit says sentence identified by ident, it's face not appears

**SayNoName(un:integer ident:string)**

Specified unit says sentence identified by ident, it's name doesn't appear - use for enemies

**ForceSay(un:integer ident:string)**

Specified unit says sentence identified by ident even if it is dying (but not dead)

**ForceSayNoName(un:integer ident:string)**

Specified unit says sentence identified by ident even if it is dying, it's name doesn't appear

**ForceSayNoFace(un:integer ident:string)**

Specified unit says sentence identified by ident even if it is dying, it's face doesn't appear

**SayRadio(un:integer ident:string)**

Specified unit says over radio sentence identified by ident, use for off map units

**SayRadioNoName(un:integer ident:string)**

Specified unit says over radio sentence identified by ident, it's name doesn't appear

**SayEffect(ident:string)**

Say effect identified by ident

**Query(ident:string)**

Displays query dialog defined in texts, returns number of answer (1..n)

**SelectiveQuery(ident:string list:plist)**

Displays query dialog defined in texts, returns number of answer (1..n)

List contains numbers of items which will be shown

**AddMedal(ident:string number:integer)**

if number>0 then adds [ident,number] item to gained\_medals

if number<0 then adds [ident,-number] item to missing\_medals

**GiveMedals(ident:string)**

No Description.

**YouWin**

No Description.

**YouLost(ident:string)**

No Description.

**YouDidSomethingExtraordinary**

No Description.

**YouWinInMultiplayer**

No Description.

**YouLostInMultiplayer**

No Description.

**MultiplayerSideAlive(side:integer)**

returns true, if there is at least one player playing this side

**SetTag(index:integer value:integer)**

sets tag[index] to value

**GetTag(index:integer)**

gets value of tag[index]

**ComMoveXY(units:plist x:integer y:integer)**

set of units receives move command to specified coordinates

**ComMoveUnit(units:plist un:integer)**

set of units receives move command to specified unit

**ComMoveToArea(units:plist area:integer)**

set of units receives move command to specified area

**ComAgressiveMove(units:plist x:integer y:integer)**

set of units receives aggressive move command to specified coordinates

**ComAttackUnit(units:plist un:integer)**

set of units receives attack command to specified unit

**ComAttackPlace(units:plist x:integer y:integer)**

set of units receives attack command to specified coordinates

**ComCollect(units:plist x:integer y:integer)**

set of units receives collect command to specified coordinates

**ComTurnXY(units:plist x:integer y:integer)**

set of units receives turn command to specified coordinates

**ComTurnUnit(units:plist un:integer)**

set of units receives turn command to specified unit

**ComEnterUnit(units:plist un:integer)**

set of units receives enter specified unit command



**ComExitVehicle(units:plist)**

set of units receives exit vehicle command

**ComExitBuilding(units:plist)**

set of units receives exit building command

**ComChangeProfession(units:plist new\_prof:integer)**

set of units receives change profession to new\_prof command

**ComResearch(units:plist techn:integer)**

set of labs receives research technology techn command

**ComConstruct(units:plist chassis:integer engine:integer control:integer weapon:integer)**

set of factories receives construct vehicle with specified components command

**ComPause(units:plist)**

set of buildings receives pause research or construction command

**ComCancel(units:plist)**

set of buildings receives cancel research or construction command

**ComHeal(units:plist un:integer)**

set of units receives heal specified unit command

**ComRepairVehicle(units:plist un:integer)**

set of units receives repair specified vehicle command

**ComRepairBuilding(units:plist un:integer)**

set of units receives repair specified building command

**ComTameXY(units:plist x:integer y:integer)**

set of units receives tame command

**ComPlaceDelayedCharge(units:plist x:integer y:integer un:integer)**

set of units receives command for place explosives to [x,y] or near unit un

**ComPlaceRemoteCharge(units:plist x:integer y:integer un:integer)**

set of units receives command for place explosives to [x,y] or near unit un

**ComFireExplosives(units:plist)**

set of units receives fire remote controlled explosives command

**ComLinkTo(units:plist hum:integer)**

set of vehicles receives link to specified human command

**ComUnlink(units:plist)**

set of vehicles receives unlink command

**ComCrawl(units:plist)**

set of units receives crawl command

**ComWalk(units:plist)**

set of units receives walk command

**ComFree(units:plist)**

set of units receives free command

**ComHold(units:plist)**

set of units receives hold command

**ComStop(units:plist)**

set of units receives stop command

**ComWait(units:plist time:integer)**

set of units receives wait command

**ComRemember(units:plist)**

set of units receives remember command

**ComReturn(units:plist)**

set of units receives return command

**ComBuild(units:plist bud:integer x:integer y:integer dir:integer)**

set of units receives build command

**ComUpgrade(units:plist)**

list of units receives command upgrade

**ComUpgradeLab(units:plist labtype:integer)**

list of units receives command upgrade to specified laboratory

**ComPlaceWeapon(units:plist weap:integer)**

list of turrets and bunkers receives place specified weapon command

**ComAnim(units:plist atype:integer)**

list of units receives do specified animation command

**ComRefuel(units:plist un:integer)**

list of units receives refuel command

**ComTransport(units:plist un:integer mattype:integer)**

list of units receives transport command

**ComInvisible(units:plist)**

list of units receives invisible command

**ComSpaceShift(units:plist x:integer y:integer)**

list of units receives command to use spaceshifting grenade

**ComTimeShift(units:plist x:integer y:integer)**

list of units receives command to use timeshifting grenade

**ComHack(units:plist un:integer)**

list of units receives hack command

**ComTeleportExit(units:plist x:integer y:integer un:integer)**

list of units receives command for place exit of teleport un to [x,y]

**ComHiddenCamera(units:plist x:integer y:integer)**

list of units receives place hidden camera

**ComContaminate(units:plist x:integer y:integer)**

list of units receives contaminate siberite deposit

**ComUnload(units:plist)**

list of units receives cargo unload command

**ComGet(units:plist x:integer y:integer)**

list of units receives simple get cargo command

**ComGive(units:plist un:integer)**

list of units receives simple give cargo command

**ComCarabine(units:plist)**

list of sheiks receives CarabineOn command

**ComSabre(units:plist)**

list of sheiks receives SabreOn command

**ComSailEvent(units:plist num:integer)**

unit raises generic event SailEvent(num)

**ComStand(units:plist)**

set of units receives stand command

**ComAttackSoporific(units:plist un:integer)**

attack unit un with soporific amunition

**ComDismantle(units:plist un:integer)**

list of units receives command to dismantle building un

**ComRecycle(units:plist un:integer)**

list of vehicles receives command to recycle in building un

**ComLinkToBase(units:plist un:integer)**

depot or warehouse receives command to link building un to the base

**ComBuildBehemoth(units:plist bud:integer x:integer y:integer dir:integer)**

set of units receives build behemoth command

**AddComMoveXY(units:plist x:integer y:integer)**

set of units receives move command to specified coordinates

**AddComMoveUnit(units:plist un:integer)**

set of units receives move command to specified unit

**AddComMoveToArea(units:plist area:integer)**

set of units receives move command to specified area

**AddComAgressiveMove(units:plist x:integer y:integer)**

set of units receives aggressive move command to specified coordinates

**AddComAttackUnit(units:plist un:integer)**

set of units receives attack command to specified unit

**AddComAttackPlace(units:plist x:integer y:integer)**

set of units receives attack command to specified coordinates

**AddComCollect(units:plist x:integer y:integer)**

set of units receives collect command to specified coordinates

**AddComTurnXY(units:plist x:integer y:integer)**

set of units receives turn command to specified coordinates

**AddComTurnUnit(units:plist un:integer)**

set of units receives turn command to specified unit

**AddComEnterUnit(units:plist un:integer)**

set of units receives enter specified unit command

**AddComExitVehicle(units:plist)**

set of units receives exit vehicle command

**AddComExitBuilding(units:plist)**

set of units receives exit building command

**AddComChangeProfession(units:plist new\_prof:integer)**

set of units receives change profession to new\_prof command

**AddComResearch(units:plist techn:integer)**

set of labs receives research technology techn command

**AddComConstruct(units:plist chassis:integer engine:integer control:integer weapon:integer)**

set of factories receives construct vehicle with specified components command

**AddComPause(units:plist)**

set of buildings receives pause research or construction command

**AddComCancel(units:plist)**

set of buildings receives cancel research or construction command

**AddComHeal(units:plist un:integer)**

set of units receives heal specified unit command

**AddComRepairVehicle(units:plist un:integer)**

set of units receives repair specified vehicle command

**AddComRepairBuilding(units:plist un:integer)**

set of units receives repair specified building command

**AddComTameXY(units:plist x:integer y:integer)**

set of units receives tame command

**AddComPlaceDelayedCharge(units:plist x:integer y:integer un:integer)**

set of units receives command for place explosives to [x,y] or near unit un

**AddComPlaceRemoteCharge(units:plist x:integer y:integer un:integer)**

set of units receives command for place explosives to [x,y] or near unit un

**AddComFireExplosives(units:plist)**

set of units receives fire remote controlled explosives command

**AddComLinkTo(units:plist hum:integer)**

set of vehicles receives link to specified human command

**AddComUnlink(units:plist)**

set of vehicles receives unlink command

**AddComCrawl(units:plist)**

set of units receives crawl command

**AddComWalk(units:plist)**

set of units receives walk command

**AddComFree(units:plist)**

set of units receives free command

**AddComHold(units:plist)**

set of units receives hold command

**AddComStop(units:plist)**

set of units receives stop command

**AddComWait(units:plist time:integer)**

set of units receives wait command

**AddComRemember(units:plist)**

set of units receives remember command

**AddComReturn(units:plist)**

set of units receives return command

**AddComBuild(units:plist bud:integer x:integer y:integer dir:integer)**

set of units receives build command

**AddComUpgrade(units:plist)**

list of units receives command upgrade

**AddComUpgradeLab(units:plist labtype:integer)**

list of units receives command upgrade to specified laboratory

**AddComPlaceWeapon(units:plist weap:integer)**

list of turrets and bunkers receives place specified weapon command

**AddComAnim(units:plist atype:integer)**

list of units receives do specified animation command

**AddComRefuel(units:plist un:integer)**

list of units receives refuel command

**AddComTransport(units:plist un:integer mattype:integer)**

list of units receives transport command

**AddComInvisible(units:plist)**

list of units receives invisible command

**AddComSpaceShift(units:plist x:integer y:integer)**

list of units receives command to use spaceshifting grenade

**AddComTimeShift(units:plist x:integer y:integer)**

list of units receives command to use timeshifting grenade

**AddComHack(units:plist un:integer)**

list of units receives hack command

**AddComTeleportExit(units:plist x:integer y:integer un:integer)**

list of units receives command for place exit of teleport un to [x,y]

**AddComHiddenCamera(units:plist x:integer y:integer)**

list of units receives place hidden camera

**AddComContaminate(units:plist x:integer y:integer)**

list of units receives contaminate siberite deposit

**AddComUnload(units:plist)**

list of units receives cargo unload command



**AddComGet(units:plist x:integer y:integer)**

list of units receives simple get cargo command

**AddComGive(units:plist un:integer)**

list of units receives simple give cargo command

**AddComCarabine(units:plist)**

list of sheiks receives CarabineOn command

**AddComSabre(units:plist)**

list of sheiks receives SabreOn command

**AddComSailEvent(units:plist num:integer)**

unit raises generic event SailEvent(num)

**AddComStand(units:plist)**

set of units receives stand command

**AddComAttackSoporific(units:plist un:integer)**

attack unit un with soporific amunition

**AddComDismantle(units:plist un:integer)**

list of units receives command to dismantle building un

**AddComRecycle(units:plist un:integer)**

list of vehicles receives command to recycle in building un

**AddComLinkToBase(units:plist un:integer)**

depot or warehouse receives command to link building un to the base

**AddComBuildBehemoth(units:plist bud:integer x:integer y:integer dir:integer)**

set of units receives build behemoth command

**SetRememberedX(un:integer value:integer)**

sets ComReturn x coordinate of a unit

**SetRememberedY(un:integer value:integer)**

sets ComReturn y coordinate of a unit

**SetDir(un:integer dir:integer)**

sets direction to an unplaced unit

**SetLives(units:plist value:integer)**

set lives of specified units to value

**SetSide(units:plist value:integer)**

set side of specified units to value

**SetSideBase(base:integer value:integer)**

//set side of complete base

**SetSkill(units:plist sk:integer value:integer)**

set skill sk of specified human units to value

**GiveSkillBonus(units:plist sk:integer value:integer)**

increases basic skill sk of units by value

**SetAttr(units:plist attr:integer value:integer)**

set attribute attr of specified human units to value

**SetFuel(units:plist percent:integer)**

sets fuel of specified combustion or solar vehicles list

**SetBLevel(units:plist blevel:integer)**

sets basic level of building(s)

**SetMark(units:plist mark:integer)**

sets mark of units (masha flag)

**TeleportExit(un:integer x:integer y:integer)**

creates exit on [x,y] for teleport un

**HiddenCamera(x:integer y:integer side:integer)**

creates hidden camera

**SetLastMission(humans:plist number:integer)**

changes last mission in which characters humans was present (nepouzivat)

**GiveMissionExperience(humans:plist number:integer)**

gives default experience for mission number to characters humans (nepouzivat)

**GetType(un:integer)**

returns type of unit (unit\_human, unit\_vehicle, unit\_building)

**GetNation(un:integer)**

//returns nation of unit

**GetIdent(un:integer)**

returns ident of specified unit (for debugging listings etc.)

**GetX(un:integer)**

returns x coordinate of a unit

**GetY(un:integer)**

returns y coordinate of a unit

**GetRememberedX(un:integer)**

returns ComReturn x coordinate of a unit

**GetRememberedY(un:integer)**

returns ComReturn y coordinate of a unit

**GetDir(un:integer)**

returns direction of a unit

**GetSide(un:integer)**

returns side of a unit

**GetLives(units:plist)**

returns average lives of specified units

**GetClass(un:integer)**

returns class of specified human unit

**GetSex(un:integer)**

returns sex of specified human unit

**GetSkill(units:plist sk:integer)**

returns average skill sk of specified human units

**GetAttr(units:plist attr:integer)**

returns average attribute attr of specified human units

**GetFuel(units:plist)**

returns average fuel of specified combustion or solar vehicles list

**GetEngine(un:integer)**

returns type of vehicle engine

**GetControl(un:integer)**

returns type of vehicle control

**GetWeapon(un:integer)**

returns type of vehicle weapon

**GetChassis(un:integer)**

returns type of vehicle chassis

**GetBType(un:integer)**

returns type of building, if un is not a building, returns -1

**GetBLevel(un:integer)**

returns level of building, if un is not a building, returns -1

**GetLabKind(un:integer which:integer)**

return kind of laboratory, which is 1 or 2

**GetBWeapon(un:integer)**

returns type of weapon (in case of turret or bunker, zero otherwise)

**GetExtPositions(un:integer)**

returns list of factory extensions positions [[x,y],[x,y],[x,y],[x,y],[x,y]]

**GetMark(un:integer)**

returns value of mark of unit un (masha flag)

**ShiftX(x:integer dir:integer dist:integer)**

returns x coordinate shifted by dist hexes in direction dir

**ShiftY(y:integer dir:integer dist:integer)**

returns x coordinate shifted by dist hexes in direction dir

**GetBase(un:integer)**

returns base of specified building (or zero, if it is not building)

**GetResourceType(base:integer mattype:integer)**

returns amount of material of specified type in specified base

**AddResourceType(base:integer mattype:integer amount:integer)**

adds resource of specified type to the specified base

**SetResourceType(base:integer mattype:integer amount:integer)**

sets resource of specified type to the specified base

**GetEnergy(base:integer)**

returns list of power values: [need,needmax,prod,prodmax]

**GetResources(base:integer)**

returns list of base resources: [cans,oil,sib]

**CanCarry(un:integer)**

returns true if unit is engineer or cargo holder

**Carry(un:integer)**

returns true if unit carries something

**CanCarryNext(un:integer)**

returns true if unit is not full

**GetResourceTypeXY(x:integer y:integer)**

returns resource type on specified hex or zero

**GetResourceAmountXY(x:integer y:integer)**

returns amount of resources on specified hex

**ChangeResourceAmountXY(x:integer y:integer n:integer)**

changes amount of existing resources on specified hex

**EraseResourceArea(ar:integer typ:integer)**

erases all resources in of specified type in specified area

**GetResourceArea(ar:integer typ:integer)**

returns all resources in of specified type in specified area

**GetCargoType(units:plist)**

returns type of cargo (mat\_none, mat\_cans, mat\_oil, mat\_siberite, mat\_multi)

**GetCargo(units:plist mattype:integer)**

returns (total) amount of mattype

**SetCargo(units:plist mattype:integer amount:integer)**

sets amount of mattype (each unit in list) returns true if (all) succeed

**AddCargo(units:plist mattype:integer amount:integer)**

adds amount of mattype (each unit in list) returns true if (all) succeed

**See(side:integer un:integer)**

returns true if side sees specified unit

**SeeXY(side:integer x:integer y:integer)**

returns how good side sees specified hex

**SeeArea(side:integer area:integer)**

returns how much hexes of specified area side sees

**GetFreeMode(un:integer)**

returns true if unit is in free mode

**GetDistUnits(un1:integer un2:integer)**

returns distance of two units, 99999 if one or both of them aren't on the map

**GetDistUnitXY(un1:integer x:integer y:integer)**

returns distance of a unit and a hex, 99999 if unit isn't on the map

**GetDistXY(x1:integer y1:integer x2:integer y2:integer)**

returns distance of two hexes

**GetDistUnitArea(un:integer area:integer)**

returns distance of a unit and the closest hex in the area (0 if it is in area)

**IsLive(un:integer)**

returns true if unit is on map and not dead

**IsDead(un:integer)**

returns true if unit is dead (lives on zero or totally destroyed)

**IsOK(un:integer)**

returns true if unit is on map and not dying

**IsDying(un:integer)**

returns true if unit is on map and dying

**IsConstructed(build:integer)**

returns nonzero number, if build is building under construction or if it is upgraded

**IsPlaced(un:integer)**

returns true if unit un is placed

**IsSelected(units:plist)**

returns true if selected

**IsAt(un:integer x:integer y:integer)**

returns true if unit is at specified coordinates

**IsInArea(un:integer area:integer)**

returns true if unit is in specified area (it can be driving or inside)

**InArea(x:integer y:integer area:integer)**

returns true if [X,Y] is in specified area

**IsInUnit(hum:integer)**

returns number of unit in which a unit is or zero

**IsDrivenBy(veh:integer)**

returns number of driver of a human controlled vehicle, zero otherwise

**IsControlledBy(veh:integer)**

returns number of driver of a remote controlled vehicle, zero otherwise

**UnitsInside(un:integer)**

returns list of units inside building (or vehicle) un

**HasTask(un:integer)**

returns true if unit has a tasks from player or SAIL

**IsBusy(un:integer)**

returns true if unit has a tasks (even its own)

**IsIdle(un:integer)**

returns true if unit does nothing

**IsTamedBy(un:integer)**

returns number of tamer of this animal, zero otherwise

**Crawls(units:plist)**



returns number of crawling units in list (0=false)

**WantsToAttack(un:integer)**

returns number of unit which is target of attack command of unit un, or zero

**Attacks(un:integer)**

returns number of unit which is currently attacked by unit un (aiming counts as attack), or zero

**GetTech(tech:integer side:integer)**

returns state of technology for side

**SetTech(tech:integer side:integer state:integer)**

sets state of technology for side

**GetRestrict(btype:integer side:integer)**

returns restriction btype for side

**SetRestrict(btype:integer side:integer state:boolean)**

sets restriction btype for side

**Researched(side:integer tech:integer)**

returns true if tech is researched (obsolete)

**GetTechProgress(tech:integer side:integer)**

returns progress of technology research for side

**SetTechProgress(tech:integer side:integer percentage:integer)**

sets progress of technology research for side

**GetWorkingProgress(units:plist)**

returns (average) progress of action of unit(s) - construction, etc.

**SetWorkingProgress(units:plist percentage:integer)**

sets progress of action of unit(s) - construction, etc. (not research!)

**PlaceSeeing(x:integer y:integer side:integer range:integer)**

places area of seeing (negative range for radar seeing)

**RemoveSeeing(x:integer y:integer side:integer)**

removes area of seeing (all matching areas)

**RevealFogArea(side:integer area:integer)**

reveals fog for specified side in specified area

**HideArea(side:integer area:integer)**

sets specified area invisible for specified side

**ShowArea(side:integer area:integer)**

reveal an area hidden by HideArea

**ResetFog(var res:tvalue)**

resets fog of war of all sides to black

**SetClass(units:plist value:integer)**

sets immediately class of units to value

**ChangeMissionObjectives(ident:string)**

change mission objectives, s is ident in Texts.txt

**HintSpec(ident:string kind:integer)**

adds hint (kind: 0-hint, 1-info, 2-error, 3-first)

**Hint(ident:string)**

adds hint

**ChangeMap(ident:string fname:string)**

changes map, ident identifies text while loading, fname is name of the map in the same directory (without path)

**ExitGame(var res:tvalue)**

exits game (debug)

**GetSideFog(side:integer)**

returns number of fog of specified side

**ChangeSideFog(side:integer fognumber:integer)**

set fog of specified side to specified fog number

**FogOff(sw:integer)**

if sw is true, fog is off

**BuildingsInProgress(side:integer)**

returns true if construction of any buildings of specified side isn't finish

**ReplaceEnvironment(x:integer y:integer envset:integer num:integer)**

replaces environment on x, y by element num from set

**RemoveEnvironment(x:integer y:integer)**

removes environment on x, y

**RemoveEnvironmentWithoutRebuild(x:integer y:integer)**

removes environment on x, y, don't count parameters, use only if normal RemoveEnviroment follows

**PlaceEnvironment(x:integer y:integer envset:integer num:integer)**

creates environment num from set envset on x, y

**PlaceRandEnvironment(x:integer y:integer envset:integer)**

creates random environment from set envset on x, y

**IsEnvironment(x:integer y:integer)**

returns true if there is an environment on x, y

**PlaceTreesToArea(area:integer sets:plist count:integer trials:integer dist:integer)**

creates count trees in area, set selects randomly from list of sets

for each tree makes some amount of trials and selects the one with less other trees within dist

note than tree in distance 1 is twice worse than tree in distance 2 etc.

**ListEnvironmentArea(area:integer)**

returns list of environments in area [[x,y,envset,num],...]

**PlaceEnvironmentList(list:plist)**

creates environments [[x,y,envset,num],[x,y,envset]...] num may not be defined

**RemoveEnvironmentArea(area:integer)**

removes all environment in area

**SiberiteRocket(x:integer y:integer n:integer list:tvalue)**

emulates n-th phase of siberite rocket explosion on list of units

**PrepareSiberiteRocket(var res:tvalue)**

prepares siberite rocket explosion

**SiberiteExplosion(x:integer y:integer)**

create siberite explosion

**GetGraphicsGamma(var res:tvalue)**

returns gamma correction from setting

**SetGraphicsGamma(gamma:integer)**

sets gamma correction of display

**GetGraphicsBrightness(var res:tvalue)**

returns gamma correction from setting

**SetGraphicsBrightness(brightness:integer)**

sets brightness of display

**PlaySoundXY(x:integer y:integer ident:string)**

plays sound with ident from coordinates x, y

**MusicIsPlaying(var res:tvalue)**

returns true if music is playing

**PrepareAttr(sex:integer)**

prepares random attributes for specified sex (or common, if zero)

**PrepareClassSkills(cl:integer level:integer)**

prepares skills for human with class cl (this skill is near specified level)

negative level means that this human is specialist

**PrepareSoldierSkills(level:integer)**

prepares skills for soldier, main skill is near specified level (uses PrepareClass)

**PrepareEngineerSkills(level:integer)**

prepares skills for engineer, main skill is near specified level (uses PrepareClass)

**PrepareMechanicSkills(level:integer)**

prepares skills for mechanic, main skill is near specified level (uses PrepareClass)

**PrepareScientistSkills(level:integer)**

prepares skills for scientist, main skill is near specified level (uses PrepareClass)

**PrepareCustomSkills(list:plist)**

prepares randomly skills for human creating, give list of four levels as a parameter:

if one of parameters is negative, then this skill counts as basic

**PrepareHuman(sex:integer cl:integer lev:integer)**

prepares parameters for a human with specified sex, class and level (0=random for sex or class)

negative level means that this human is specialist

**PrepareSoldier(sex:integer lev:integer)**

prepares parameters for a soldier with specified sex and level (0=random for sex)

**PrepareEngineer(sex:integer lev:integer)**

prepares parameters for a engineer with specified sex and level (0=random for sex)

**PrepareMechanic(sex:integer lev:integer)**

prepares parameters for a mechanic with specified sex and level (0=random for sex)

**PrepareScientist(sex:integer lev:integer)**

prepares parameters for a scientist with specified sex and level (0=random for sex)

**WaitForMc(mcid:integer)**

waits for finishing of macro mcid

**ExistMc(mcid:integer)**

if macro mcid exists, returns true, otherwise false

**KillMc(mcid:integer)**

if macro mcid exists, closes macro and returns true otherwise returns false

**GetUnitMc(un:integer)**

returns mcid of macro which uses unit un returns zero if not used

**GetMcUnits(mcid:integer)**

returns list of units of the macro mcid

**AddMcUnits(mcid:integer units:plist)**

changes list of units of the macro mcid returns true if successful

**RemoveMcUnits(mcid:integer units:plist)**

changes list of units of the macro mcid returns true if successful

**ClearMcUnits(mcid:integer)**

clears list of units of the macro mcid returns true if successful

**GetMcUnitsSpec(mcid:integer spec:integer)**

returns spec list of units of the macro mcid

**AddMcUnitsSpec(mcid:integer units:plist spec:integer)**

changes spec list of units of the macro mcid returns true if successful

**RemoveMcUnitsSpec(mcid:integer units:plist spec:integer)**

changes spec list of units of the macro mcid returns true if successful

**ClearMcUnitsSpec(mcid:integer spec:integer)**

clears spec list of units of the macro mcid returns true if successful

**McBase(prior:integer units:plist build:plist research:plist construct:plist options:plist)**

runs macro and returns mcid

build must be BuildList, researchs - list of techs, construct - not implemented

**McMove(prior:integer units:plist waypoints:plist options:plist)**

runs macro and returns mcid

units - list of units, what else...

waypoints - as expected...

options as follows:

mc\_move\_wait - wait for each other after 4 hexes, can take a distance in hexes as parameter

mc\_move\_through - really walk THROUGH the waypoints, default is coming 6 hexes to them

mc\_move\_waitwp - wait on waypoints for the rest of units

mc\_move\_dontshoot - dont shoot at unfriendly obstacles

mc\_move\_dontcapture - dont capture vehicles

mc\_move\_agressive - be !#\$% AGGRESSIVE !!!

mc\_move\_ownstuck - don't timeout with obstacles of own side at nervosity 100, just do it FOREVER...

### **McRegistry(side:integer options:plist)**

Runs macro for automatic registration of enemies of side in specified intervals and returns mcid.

Every registration gives count and strength of all registered units as its result. If registration returns different result than a previous (or nonzero in first registration), it raises event:

McRegistrationResultChanges(mcid,count,strength).

Value of strength has no real meaning, use it only for comparison with another result of macro or for comparison with zero.

Possible options (note that defarea can be given in two formats, as 'areanumber' or as '[x,y,r]'):

[mc\_reg\_refresh\_time, ticks] ... interval between two registrations (default 0\$1)  
[mc\_reg\_area\_to\_guard, defarea] ... registries enemies in defined area (default none)  
[mc\_reg\_area\_to\_protect, defarea] ... registries enemies attacking units in specified area (default none)  
[mc\_reg\_units\_to\_protect, listofunits] ... registries enemies attackign specified units (default none)  
[mc\_reg\_units\_to\_guard, listofunits] ... registries specified enemies (default none)  
[mc\_reg\_expire\_stops\_to\_attack, ticks] ... how long after enemy stops to attack the registration expires (default 0\$3)  
[mc\_reg\_expire\_leaves\_area, ticks] ... how long after enemy leaves area the registration expires (default 0\$2)  
(-)mc\_reg\_ignore\_fog ... ignores fog so you can see all enemies in area (default false)  
(-)mc\_reg\_only\_important ... doesn't raise event if result changes from nonzero value to another nonzero value (default false)

### **McRegistryChangeOptions(mcid:integer options:plist)**

changes options of McRegistry with ident mcid

### **McDefend(prior:integer regid:integer units:plist options:plist)**

runs McDefend macro and returns mcid

There must be running McRegistry macro to start McDefend - enemies are registred by this McRegistry.

McDefend starts after each registration of this McRegistry. Parameters:

[mc\_def\_advantage, percent] ... how great advantage over enemy is enough (default 200, twice)

[mc\_area\_dont\_leave, defarea] ... area where must all defenders be (default none)

(-)mc\_change\_to\_soldiers ... set here if McDefend can switch people to soldiers

(-)mc\_change\_to\_mechanics ... set here if McDefend can switch people to mechanics

### **McAttack(prior:integer regid:integer units:plist options:plist)**

runs McAttack macro and returns mcid

There must be running McRegistry macro to start McAttack - enemies are registred by this McRegistry.

McAttack starts after each registration of this McRegistry.

regid - registry id - the units watch the area\_to\_guard from mc\_registry

units - list of units obeying your commands, o designer master...

options:

[-]mc\_no\_stop ... crawling throug the registered area forever

[-]mc\_murder ... kill the DAMNED bastards lying on the ground !

[mc\_area\_dont\_leave, defarea] ... dont leave this area (default none)

[mc\_retreat\_lives\_people,lives] ... retreat to retreat\_area\_people when (wo)man's hitpoints get under lives

[mc\_retreat\_lives\_vehicles,lives] ... retreat to retreat\_area\_vehicle when vehicle's hitpoints get under lives

[mc\_retreat\_area\_people, defarea]

[mc\_retreat\_area\_vehicle, defarea]

[mc\_out\_of\_fuel, level] ... throw an event when fuel lowers under level

Behaviour in Czech - sorry you english-only guys (ang girls):

Chovani McAttacku:

Chodi po zadane oblasti, dokud neuvidi nepratele.

Tem pak prirazuje Nevhodnost\_cilu pro stranu McAttacku, nejmensi pro nejslabsi jednotky.

Nevhodnost je normalizovana hodnotou 100.

Jakmile je vsechny pobije a neuvidi zadne dalsi, skonci, pokud nema nastaveny flag no\_stop.

Da se mu nastavit oblat, kterou nema opoustet a oblasti, kam se maji stahovat zranene jednotky.

Haze event u jednotek, kterym dochazi palivo.

Vytvari si seznam jednotek(cilu) na ktere prave utoci a pri pristim pruchodu jim nastavi nevhodnost cilu na -1.

tim se zajisti, ze nedorazi umirajici (kterym jinak zustane velmi prizniva Nevhodnost\_cile).



Kdyz McAttack skonci, nastavi vsem jednotkam na ktere utocil nevhodnost -1.

Standardne McAttack nedorazi mrtve. Da se k tomu premluvit zapnutim prepinae Murder McAttack, kdy prestane nastavovat nevhodnost umirajicim na -1 a ti pak maji nizkou nevyhodnost.

Po ukonceni McAttacku temto jednotkam zustane nastavena nevhodnost, takze je kdokoliv kolemjdouci patrici k nasi strane dorazi.

Nenapada me jednoduchy zpusob jak toto vyresit (tj. aby i umirajici jednotky v tomto rezimu mely po skonceni McAttack nevhodnost -1).

McAttack skonci, pokud jiz nema zadne jednotky, ktere by mohly utocit, nebo pokud nevidi zadne neumirajici nepratelske jednotky.

Po skonceni McAttacku jednotky nemaji zadny ukol a tudiz dorazi vsechny nepratele, ktere vidi.

### **McPatrol(prior:integer regid:integer units:plist options:plist)**

runs McPatrol macro and returns mcid

There must be running McRegistry macro to start McPatrol - enemies are registred by this McRegistry.

McPatrol starts after each registration of this McRegistry.

units walk through an area and try to visit parts they havent seen for a long time and they do it FOREVER

(ie. McPatrol ends iif all units in it are dead

throws an event as soon as the units see an enemy

almost all options just like in McAttack, except mc\_murder and mc\_no\_stop, which are missing. One more is:

[-]mc\_pat\_aggresive ... be agresive while patrolling

### **McSkirmish(prior:integer units:plist options:plist)**

runs McSkirmish macro and returns mcid

options - [[mcs\_PosList, list], [mcs\_Strategy, list], [mcs\_Parking, parkingarea]]

### **McSSetPosList(mcid:integer list:plist)**

sets building position list for McSkirmish mcid

### **McSAddVehicles(mcid:integer weapons:plist chassis:plist engines:plist controls:plist parkingarea:integer)**

creates new vehicle definition for McSkirmish mcid, returns number of group

### **McSChangeVehicles(mcid:integer number:integer weapons:plist chassis:plist engines:plist controls:plist parkingarea:integer)**

updates vehicle definition for McSkirmish mcid, group number

**McSAddFortification(mcid:integer kinds:plist weapons:plist)**

create new fortification definition for McSkirmish mcid, returns number of definition

**McSChangeFortification(mcid:integer number:integer kinds:plist weapons:plist)**

updates fortification definition for McSkirmish mcid, definition number

**McSSetStrategy(mcid:integer list:plist)**

sets main strategy list for McSkirmish mcid

**McSGetVehList(mcid:integer number:integer)**

returns list of vehicles in group number

**McSParking(mcid:integer veh:plist park:boolean)**

changes list of parking vehicles veh will be added (park=true) or removed (park=false)

**McSConditionalStrategyFilter(str:plist)**

filters conditional strategy to strategy without conditions (main strategy list)

**McRepair(prior:integer units:plist others:plist options:plist)**

runs McRepair macro and returns mcid

units will be maintain others (and extras from McWithdraw)

options - [[mcr\_hum\_area, area], [mcr\_veh\_area, area], (-)mcr\_area\_only, [mcr\_safety, 7],

[mcr\_hum\_limit, 600], [mcr\_veh\_limit, 1000], [mcr\_bui\_limit, 1000],

[mcr\_repv\_area, area], [mcr\_capv\_area, area], [mcr\_repb\_area, area], [mcr\_capb\_area, area]]

**McWithdraw(prior:integer repid:integer units:plist options:plist)**

runs McWithdraw macro and returns mcid there should be running McRepair first

options - [[mcw\_hum\_lo, 400], [mcw\_hum\_hi, 800], [mcw\_veh\_lo, 400], [mcw\_veh\_hi, 1000],

[mcw\_hum\_area, defarea], [mcw\_veh\_area, defarea], [mcw\_corpse\_area, defarea]]

areas will be copied from McRepair of repid before evaluating this options

**McCustom(prior:integer units:plist)**

runs void macro and returns mcid

**GetBuildList(base\_or\_list:tvalue)**

creates BuildList from whole base or list of units (buildings)

**SetBattleFlag(side:integer x:integer y:integer options:plist)**

sets battle flag for specified side to specified coordinates and returns flagid

effect of flag depends on its options, default options:

```
[[bfo_defend_bonus_human,80],[bfo_defend_bonus_vehicle,40],[bfo_defend_bonus_building,0],[bfo_range,17],[bfo_height,100]]
```

distance: 0..range: full effect range+1..range\*2: decreasing effect

### **KillBattleFlag(flagid:integer)**

removes battle flag flagid (returns true if successful)

### **FindBattleFlag(x:integer y:integer)**

returns flagid of battle flag on [x,y], or -1

### **GetBattleFlagSide(flagid:integer)**

returns side of existing battle flag

### **ChangeBattleFlagSide(flagid:integer side:integer)**

changes side of existing battle flag

### **ChangeBattleFlagOptions(flagid:integer options:plist)**

changes options of existing battle flag

### **SetAreaMapShow(area:integer id:integer)**

shows area on map and/or minimap

### **MoveAreaXY(area:integer x:integer y:integer)**

moves area, left upper corner of deltoid is at specified coordinates

### **GetMultiplayerSetting(index:integer)**

returns value of multiplayer setting with specified index

### **RaiseSailEvent(num:integer)**

raises immediately generic event SailEvent(num)

### **HexInfo(x:integer y:integer)**

returns number of unit on hex, 0 for free and -1 for environment

### **SendSiberiteRocket(x:integer y:integer)**

siberite rocket will explode at hex x,y

**CanBeResearched(labor:integer tech:integer)**

returns if tech can be researched in lab (if tech is already researched, returns false)

**PlaceWeaponTurret(un:integer bc\_weapon:integer)**

place bc\_weapon on turret or bunker un

**UnitsLinked(un:integer)**

returns number of linked vehicles on unit

**FindMaxSkill(units:plist skill:integer)**

returns unit with max skill from units

**FindMaxSkill2(units:plist skill:integer)**

returns unit with max skill from units (including nonplaced) added by angry jam

**GetListOfCratesInArea(area:integer)**

returns list of crates positions in area

**GetUnitActivity(un:integer)**

returns units activity (constants in UDatStr.pas - cin...)

**GetTaskList(un:integer)**

returns list of lists [task,x,y,obj,par1,par2,patrol] siiiib (UDatStr - ukol...)

**GetUnitNumber(x:integer y:integer)**

returns number of unit at pos x, y

**FindNearestPoint(list\_of\_points:plist x:integer y:integer)**

returns nearest point to x y from list\_of\_points

**GetResourceVisibility(x:integer y:integer side:integer)**

returns true, if deposit at pos x, y is visible for side

**SetResourceVisibility(x:integer y:integer side:integer)**

sets deposit at pos x, y visible for side

**GetListOfResourcesInArea(area:integer)**

returns list of deposits positions according to the type, [x,y,typ,...]

**FindMinSkill(units:plist skill:integer)**

returns unit with min skill from units

**CopyTasks(un\_dest:integer un\_source:integer)**

copy tasks from unit un\_source to unit\_dest (replacing, no patrol)

**AddCopyTasks(un\_dest:integer un\_source:integer)**

copy tasks from unit un\_source to unit\_dest (adding, no patrol)

**SetTaskList(un:integer list:plist)**

sets tasks from tasklist (GetTaskList format) - use carefully!

**AddTaskList(un:integer list:plist)**

adds tasks from tasklist (GetTaskList format) - use carefully!

**CanBeConstructed(factory:integer chassis:integer engine:integer control:integer weapon:integer)**

returns if vehicle can be constructed in factory

**CostOfVehicle(chassis:integer engine:integer control:integer weapon:integer)**

returns cost in list [cans,oil,sib]

**CostOfBuilding(kind:integer nation:integer)**

returns cost in list [cans,oil,sib]

**CostOfWeapon(weapon:integer)**

returns cost in list [cans,oil,sib]

**SeeGreyArea(side:integer area:integer)**

returns how much hexes of specified area side sees + gray hexes

**MineExplosion(x:integer y:integer wound:integer)**

explosion of mine at x, y

**PlaceMine(x:integer y:integer side:integer typ:integer)**

place mine at pos x, y. typ=0 - remote mine (result number of mine), typ>0 (timed mine - time in tick (1..100))

**LaunchMine(number:integer)**

explosion of remote mine, that was placed by PlaceMine Rumun: DO NOT USE number, it changes

**LaunchMineAtPos(x:integer y:integer side:integer)**

launch mine at pos x, y, that was placed by PlaceMine

**ViewMineAtPos(x:integer y:integer side:integer)**

reveal mine at pos x, y for side\_fog[side]

**MineAtPos(x:integer y:integer)**

returns true, if it is mine at pos x, y otherwise false

**MineOfUnit(un:integer)**

returns [x,y], if unit has placed remote charge otherwise returns false

**RemoveMineOfUnit(un:integer)**

removes placed remote charge of unit, if exists

**BuildingStatus(un:integer)**

returns building status (bs\_???)

**ContactTime(sides:plist)**

time from last contact (parameter: side, or [side1,side2])

**InBattle(sides:plist)**

battle detection (parameter: side, or [side1,side2])

**GetProperties(un:integer)**

return list of unit properties [speed, defend, sight, range, eff1, eff2, eff3]

**ImagineProperties(vehb:integer human:integer cl:integer)**

return list of unit properties for vehb with human inside (cl 0 means actual class)

**CanCarryHowMuch(un:integer)**

returns how much material can unit still carry

**SetArtifactRes(side:integer state:boolean)**

allows researching of artifact technology

**SetArtifactUse(side:integer number:integer state:integer lab:integer)**

allows use of artifact icon in labs of side...

number - art\_use\_? state - art\_? lab - unit number, negative kind, or zero for all labs

**FindArtifact(size:integer)**

finds artifact with size size and returns its position [x,y], or false

**PriorityAttack(side:integer un:integer)**

prioritni utok strany side na jednotku un

**DoNotAttack(side:integer un:integer)**

neutoceni pro stranu side na jednotku un

**NormalAttack(side:integer un:integer)**

zruseni speciality pro stranu side na jednotku un

**EnableExclamations(var res:tvalue)**

exclamations enabled

**DisableExclamations(var res:tvalue)**

exclamations disabled

**AvailableChassisList(un:integer)**

returns list of available components (chassis) - factory, workshop

**AvailableEngineList(un:integer)**

returns list of available components (engines) - factory, workshop

**AvailableControlList(un:integer)**

returns list of available components (controls) - factory, workshop

**AvailableWeaponList(un:integer)**

returns list of available components (weapons) - factory, workshop, turret

**GetBuildingTechReq(kind:integer nation:integer)**

returns technology required to building of kind, nation (0 - free -1 - not available to nation)

**GetTechTechsReq(tech:integer)**

returns list of technologies required to research tech

**GetTechLab(tech:integer)**

returns kind of lab required to research tech

**TechNationAvailable(tech:integer nation:integer)**

returns true if tech is available to nation

**TeleportUnit(un:integer x:integer y:integer range:integer sound:boolean)**

try to teleport unit un to [x,y]. Uses warp effect. Sound optionally. Returns true, if succeed.

Only for outside units. Use range to auto-find other destination if [x,y] unusable.

**GetCustomRate(basicrate:integer tab:integer skill:integer)**

returns basicrate bonused (or penalized) by skill (-10..50) using skilltable tab (0..6)

Usually used skills are 0..10. Table 0 gives no bonus, table 6 gives biggest bonus.

e.g. scientist's research rate is GetCustomRate(zaklad,4,GetSkill(human,skill\_scientistic))

**GetActResearch(un:integer)**

returns technology actually researching in lab un 0 if no or spec research

**SetSpecResearch(un:integer time:integer auto:boolean)**

sets (laboratory) un to SpecResearch auto mode time - length use carefully!

**SetNoActivity(un:integer)**

resets activity of (laboratory) un use VERY carefully!

**ValidHex(x:integer y:integer)**

{true if hex x,y is valid} 0 - invalid, 1 - bordering, 2 - normal.

**Contaminate(side:integer x:integer y:integer)**

immediatelly contaminates siberite deposit if succesfull, (raises event and) returns true



**SelectUnits(units:plist)**

add selection to units

**DeselectUnits(units:plist)**

removes selection of units

**AddExperience(hum:integer sk:integer points:integer)**

adds experience points in skill sk to human hum

**RemoveTasks(units:plist)**

removes all tasks of units

**EffectTeleport(x:integer y:integer)**

makes effect on hex [x,y]

**ArtContamination(x:integer y:integer percentage:integer)**

makes contaminated area (percentage of siberite rocket contamination)

**ComForceInvisible(units:plist)**

list of units receives invisible command

**AddComForceInvisible(units:plist)**

list of units receives invisible command

**EnableVideoExclamations(var res:tvalue)**

video exclamations enabled

**DisableVideoExclamations(var res:tvalue)**

video exclamations disabled

**SetBName(units:plist bname:string)**

sets building's display string

**AreaToMatrix(area:integer)**

creates a matrix from the area

**WaypointsFromMatrix(matrix:plist areas:plist)**

creates waypoints through areas using matrix. Areas is a list:

[[x,y,r],...] and there must be at least one item in this list.

The way is randomly created: random points in the areas are selected, then one of the best ways through these points is randomly selected

**SideShoot(un:integer)**

returns side, that last shoot unit un

**UnitShoot(un:integer)**

returns unit, that last shoot unit un

**SetUnitDisplayNumber(un:integer num:integer)**

displays number num over the unit un (even in final Original War)... zero means display nothing

**SetMultiScore(side:integer score:integer)**

sets multiplayer score of specified side

**GetMultiScore(side:integer)**

gets multiplayer score of specified side

**ResetMultiScore(var res:tvalue)**

resets multiplayer score for all sides

**Constructable(chas:integer eng:integer con:integer weap:integer)**

returns true, if vehicle constructable

**CopySkills(un1:integer un2:integer)**

copies skills between units (un1->un2)

## \* Zakończenie

### **Podsumowanie:**

W ten sposób doszliśmy do końca kursu, który przedstawiał podstawy tworzenia modów, do gry "Original War". Kurs opisuje tylko te najważniejsze funkcje *Edytora*, bowiem część z nich z oczywistych powodów została ominięta.

Aktualnie na scenie "Original War", jest niewiele takich kursów, lecz mam nadzieję, iż w przyszłości będzie ich więcej, oraz że więcej osób będzie zajmowało się tworzeniem modów do tej gry.

### **Prawa Autorskie:**

Niniejszy kurs, w tym jego struktura, a w szczególności teksty i zdjęcia stanowią wyłączną własność Daniela Gabryśia oraz podlegają ochronie zgodnie z ustawą z dnia 4 lutego 1994 roku o prawie autorskim i prawach pokrewnych (tj. Dz. U. z 2000r., Nr 80, poz. 904, z późn. zm.). Wszelkie teksty autorstwa innych osób, dodane do kursu jako rozwinięcie kursu (tj. spis funkcji saila spisany przez firmę deweloperską *Altar Interactive*), są własnością ich autorów.

Żadna część tego kursu, łącznie z tekstami, zdjęciami, grafiką oraz znaki towarowe, nie może być kopiowana ani rozpowszechniana, w tym również w celu wykorzystania w całości lub w części w innych kursach, w publikacjach elektronicznych jak również w wersji materialnej, bez pisemnej zgody.