

SRF Engine Help

W polskiej wersji językowej

Autor instrukcji i silnika: Sebastian Kąkolewski (Seka's Studio)

Do czego służy i czym właściwie jest SRF Engine?

SRF Engine to silnik pozwalający odczytywać i wykonywać pliki *.SRF (Pliki o prostym formatowaniu). Pozwalają one na szybkie rysowanie tekstów za pomocą stylów. Do zalet tego rozwiązania oprócz obsługi stylów należy szybkie wykonywanie nowych linii. Ilość zastosowań i komend będzie oczywiście rozwijana.

Czym są style?

Style są formatami(czcionką, kolorem), które można szybko zmieniać za pomocą komendy:

```
[s=Style_name]
```

Przy czym zamiast Style_name wpisujemy nazwę naszego stylu.

Jak powinien wyglądać plik *.SRF i jak go utworzyć?

Plik *.SRF można edytować za pomocą zwykłego notatnika dołączonego do systemu windows. Plik tworzy się zapisując projekt w notatniku używając opcji „Zapisz Jako...”, wybieramy zapisz jako typ „Wszystkie pliki (*)” i zapisujemy po nazwie dodając rozszerzenie „.SRF”. Pliki takie powinny być dodane w projekcie do „Included Files”(dotyczy tylko Game Maker Studio, w GM 8/8.1 nie ma takiej potrzeby, ale może być potrzebne uruchomienie programu jako administrator). Plik powinien zawierać zwykły tekst, jedynie ze znacznikami zmiany stylów np.:

```
[s=Head]Hello World
```

```
[s=Normal]Hello World
```

Oczywiście musimy najpierw utworzyć używane style, po za plikiem(w kodzie aplikacji). W tym wypadku należy stworzyć style „Head” i „Normal”.

Jak użyć SRF Engine?

SRF Engine wymaga najpierw inicjacji(wystarczy jeden raz na aplikację, następne użycia go zresetują).

W tym celu używamy funkcji:

```
SRF_initiation(debug_msgs);
```

zamiast debug_msgs należy wpisać:

```
true
```

jeżeli chcemy uruchomić silnik w trybie debugowania(w tym przypadku tracimy na wydajności, ale mamy doskonały podgląd działania silnika), lub:

```
false
```

jeżeli chcemy uruchomić silnik z wyłączonym trybem debugowania(zalecane przede wszystkim w ostatecznej wersji). Następnie tworzymy style, których będziemy używać do tego służy funkcja:

```
SRF_style_create(font, color, name);
```

przy czym zamiast font wpisujemy czcionkę dla tworzonego stylu, zamiast color kolor dla tworzonego stylu, a zamiast name – nazwę dla tworzonego stylu. Funkcja ta zwraca id stylu.

Teraz wykonujemy plik SRF. W tym wypadku mamy dwie możliwości.

Funkcja zalecana tylko przy jednorazowym użytku pliku:

```
SRF_execute_file(free_bufor_id, x, y, file);
```

(zwraca id utworzonego stylu)gdzie zamiast free_bufor_id wpisujemy id wolnego bufora(od 0-200), jeżeli chcemy jakiś zajęty(można i wolny, ale to zbędne) bufor wyczyścić, używamy:

```
SRF_bufor_clear(bufor_id);
```

zamiast bufor_id wpisujemy id czyszczonego bufora, zamiast x i y wpisujemy pozycję, gdzie chcemy narysować treść pliku, oraz za file wpisujemy nazwę/adres pliku.

Pamiętajmy, aby wyczyścić bufor, i usunąć zbędne style jest to pomocne przy optymalizacji.

Do tego służą funkcje:

```
SRF_bufor_clear(bufor_id); //czyści bufor
```

```
SRF_style_delete(idStyle); //usuwa style
```

Druga funkcja zalecana jeżeli używamy tego samego pliku często, cykliczna. Wtedy nie musimy za każdym razem wczytywać od nowa pliku:

```
SRF_execute_bufor(bufor_id, x, y);
```

argumentacja jest podobna do funkcji pierwszej, wykonuje wcześniej wczytany do bufora plik(plik wczytujemy za pomocą funkcji:

```
SRF_load(file, bufor_id);
```

bufor_id to id bufora do którego wczytujemy plik, a file to adres/nazwa pliku.

Przykład użycia

Kod create obiektu kontrolera:

```
///Inicjacja SRF
globalvar stHead, stNormal, stSpace, stHead2; //Wskaźnik do naszego stylu
SRF_initiation(true); //inicjuje silnik SRF
stNormal=SRF_style_create(font1, merge_color(c_green, c_white, 0.2), "Normal"); //Tworzy styl normalny i przypisuje
jego id do wskaźnika
stHead=SRF_style_create(font0, c_green, "Head"); //Tworzy styl nagłówka i przypisuje jego id do wskaźnika
stSpace=SRF_style_create(font2, c_green, "Space"); //Tworzy styl odstępi między nagłówkiem, a tekstem i przypisuje
jego id do wskaźnika

stHead2=SRF_style_duplicate(stHead, "Head2"); //Tworzy styl nagłówek2 na podstawie stylu Head
SRF_style_modification(stHead2, SRF_style_get_font(stNormal), SRF_style_get_color(stHead),
SRF_style_get_name(stHead2)); //zmienia parametry stylu Head2 (czcionkę na tą z stylu Normal i kolor na ten ze
stylu o Head), a nazwę zostawia starą
SRF_load("simple_text.SRF", 0); //Wczytuje do bufora plik
```

Room end:

```
///Usuwanie zbędnych danych SRF
SRF_bufor_clear(0); //czyści bufor o id 0
//Usuwa style:
SRF_style_delete(stNormal);
SRF_style_delete(stHead);
SRF_style_delete(stHead2);
SRF_style_delete(stSpace);
```

Draw:

```
///Rysuj SRF
SRF_execute_bufor(0, 10, 10); //Wykonaj plik SRF z bufora 0
```

Treść pliku simple_text.SRF:

```
[s=Head]SRF File Opener by Sebastian Kąkolewski[s=Head2](Seka's Studio)
[s=Space]
[s=Normal]Author this engine is Sebastian Sebastian Kąkolewski(Seka Studio)
Please add to your project with engine, info about enigne's author:)
```

Pozostałe funkcje

```
SRF_exist(style);
```

Sprawdza czy style o podanym id(**pierwszy argument to nie nazwa stylu tylko id**).

```
SRF_style_set(id_style);
```

Ustawia dany styl. Jeżeli chcemy kodem, który jest poza plikiem SRF zmienić styl, najlepiej użyć tej funkcji. Istnieje też drugi sposób(ale mniej optymalny):

```
SRF_execute_commend(commend);
```

przy argumencie commend wpisujemy „[s=nazwa_stylu]”.

```
SRF_find_style_with_name(style_name);
```

Szuka stylu o danej nazwie. Zwraca jego id o ile taki znajdzie, w przeciwnym wypadku zwraca 0.

Licencja, warunki użycia

Silnika można używać w każdej produkcji, nawet komercyjnej z zaznaczeniem, że użyliśmy dany silnik, danego autora(jeżeli zmodyfikowaliśmy silnik na własne potrzeby, ew.: dopisujemy siebie jako moderatora silnika). Wszelkie modyfikacje silnika, i publikacje w zmodyfikowanej wersji są dozwolone, ale z zaznaczeniem pierwotnego autora silnika(ew.: przy zmianie nazwy silnika, pierwotną nazwę).