Krzysztof Łukasz Necel, 143301

Tomasz Gajger, 143218

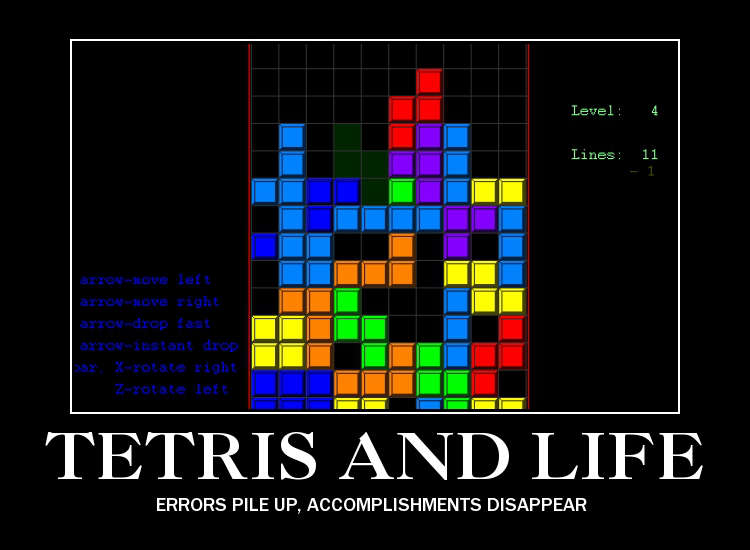
**Oprogramowanie Systemowe Tetris w EFI**

**Wstęp**

Coś tam, coś tam, coś tam...

**Tematyka pracy**

Tematem pracy jest stworzenie gry podobnej do popularnego Tetrisa[[1]](#footnote-1), ale działającej w środowisku shella EFI. Projekt będzie realizowany w oparciu o środowisko EDK II - EFI Developer Kit II[[2]](#footnote-2).

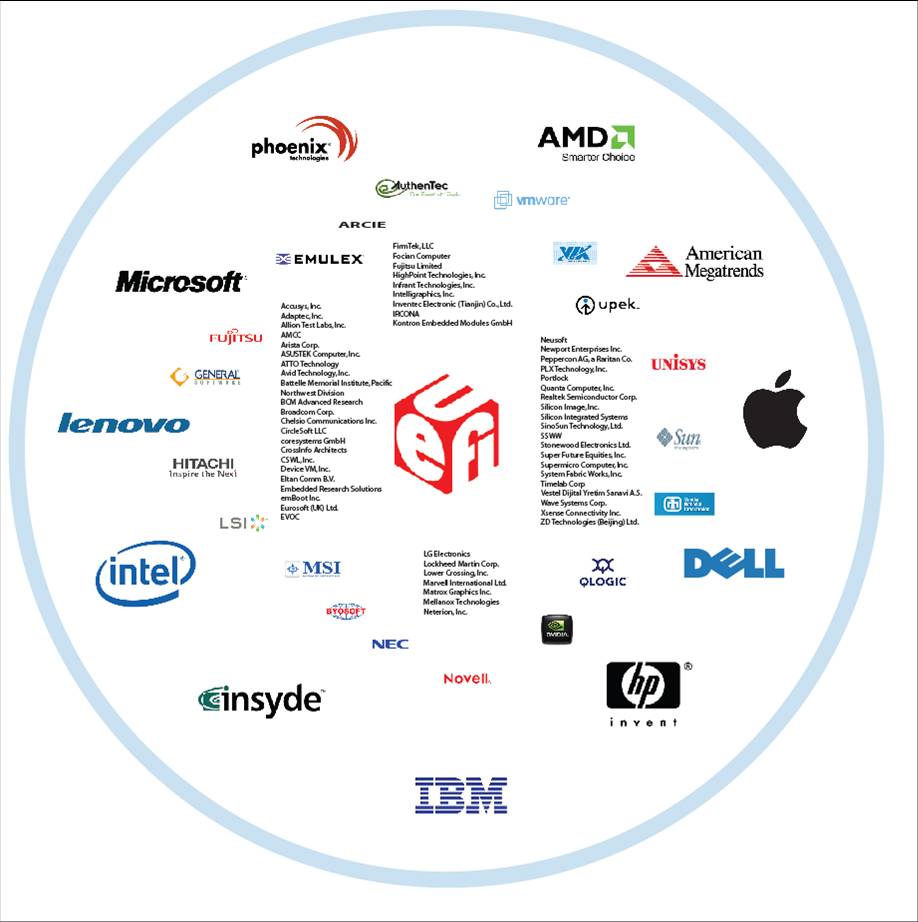


Celem, który chcemy osiągnąć, jest umieszczenie naszej gry w obrazie ISO[[3]](#footnote-3), zamontowanie go w Virtual Boxie[[4]](#footnote-4) i zaprezentowanie jej działania na maszynie wirtualnej.

**Przegląd technologii**

Unified EFI Forum jest organizacją handlową non-profit utworzoną w celu promowania i zarządzania standardem UEFI. Jako, że standard ten jest dynamicznie rozwijającym się, tworzenie jego specyfikacji jest kierowane i wspierane przez firmy będące członkami Forum UEFI.

Zarząd Forum UEFI stanowią reprezentanci jedenastu czołowych firm: AMD, American Megatrends Inc., Apple Computer Inc., Dell, Hewlett Packard, IBM, Insyde, Intel, Lenovo, Microsoft, Phoenix Technologies.



**Często zadawane pytanie: Jaki jest związek pomiędzy EFI, a UEFI?**

**Odpowiedź:** UEFI bazować będzie na specyfikacji EFI 1.10, opublikowanej przez Intel, z poprawkami i zmianami zarządzanymi przez Forum UEFI. Intel posiada *Copyright* na specyfikację EFI 1.10, ale przekazał ją do użytku Forum, tak aby Forum mogło ją rozwijać. Nie będzie kolejnych wersji specyfikacji EFI, specyfikacja UEFI zostanie wydana przez Forum, nie przez Intela.[[5]](#footnote-5)

EFI oraz programy uruchamiane na nim działają zanim jeszcze system zostanie załadowany, środowisko to jest idealne dla pewnych typów programów, dobre dla innych, a kompletnie wyklucza jeszcze inne. W związku z tym, programy napisane pod EFI mogą być uruchamiane błyskawicznie po włączeniu zasilania komputera, środowisko to jest również pozbawione wielu komplikacji, które wnoszą ze sobą systemy operacyjne, mamy w nim, na przykład, bezpośredni dostęp do sprzętu.[[6]](#footnote-6)

Rozważmy następujące typy programów:

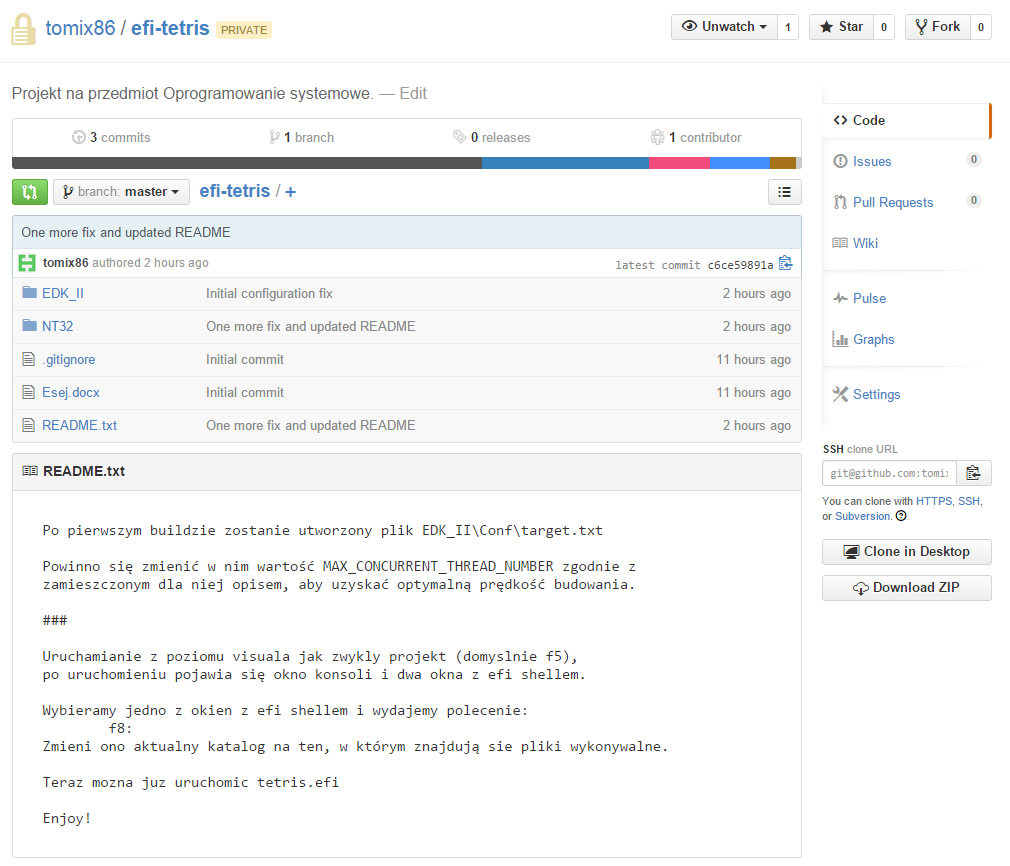
* **Boot managery** — programy służące do wyświetlania użytkownikowi menu, z którego może wybrać dostępne opcje bootowania. Programy takie są prawie zawsze pisane tak, aby były uruchamiane zanim system zostanie załadowany, a kontrola przekazana do jego jądra. Na komputerze z EFI, program taki pisany jest pod środowisko EFI.
* **Boot loadery** — programy, których zadaniem jest podjęcie kroków niezbędnych, aby załadować jądro systemowe do pamięci i zainicjalizować jego wykonanie.
* **Narzędzia do konfigurowania/wgrywania firmware'u** — Przez lata użytkownicy komputerów PC musieli korzystać z prostych, topornych i sprawiających wrażenie bardziej skomplikowanych niż w rzeczywistości są BIOSów. Wraz z wprowadzeniem EFI możliwości programistów zostały znacznie poszerzone, obecnie, posiadając wystarczającą wiedzę i umiejętności, można stworzyć interfejsy z rozbudowanym GUI, obsługą myszy i innymi przydatnymi funkcjonalnościami. EFI shell jest przykładem takiego interfejsu o prostej budowie.
* **Sterowniki EFI** — Programy te dają EFI dostęp do sprzętu czy systemu plików. Często stanowią one część firmware'u płyt głównych lub innych urządzeń, ale mogą być załadowane podczas procesu bootowania, tak aby za ich pośrednictwem programy rozruchowe mogły korzystać z szerszej gamy funkcji oferowanych przez sprzęt.
* **Narzędzia do diagnostyki sprzętu** — narzędzia sprawdzające pamięć RAM, procesor czy dysk twardy mogą również działać w środowisku EFI. Dzięki temu istnieje możliwość wykorzystania ich nawet w przypadku awarii systemu operacyjnego.

Przykłady programów z powyższych kategorii można z łatwością znaleźć, niektóre programy można nawet zaklasyfikować do kilku z nich, na przykład, GRUB 2 jest zarówno boot managerem jak i boot loaderem. EFI shell z kolei można uznać za program do konfigurowania firmware'u oraz pewnego rodzaju program użytkowy.

Widać wyraźnie, że potencjał EFI jako platformy o szerokiej gamie zastosowań, czeka wciąż na pełne odkrycie. Wraz z biegiem czasu, naszym oczom, powinny się ukazywać coraz bardziej przydatne, kreatywne i złożone programy.

**Przegląd środowiska**

Wymianę plików prowadzić będziemy za pomocą GitHuba.



Wstępna konfiguracja środowiska polegała na pobraniu źródeł[[7]](#footnote-7) EDK2 i próbie ich budowy z poziomu konsoli[[8]](#footnote-8), w tym miejscu wystąpiły drobne problemy, gdyż VS 2013 potraktował pewne ostrzeżenia poziomu /W4 jako błędy, co uniemożliwiło kompilację. Musiałem więc dokonać ręcznej modyfikacji dwóch plików źródłowych, wyłączając ostrzeżenia[[9]](#footnote-9),[[10]](#footnote-10) dyrektywą

*#pragma warning (disable: 4701 4703)*

Po wykonaniu tego zabiegu kompilacja przebiegła pomyślnie.

Następnie skonfigurowałem[[11]](#footnote-11) Visual Studio 2013, które wykorzystamy jako IDE, tak aby pozwalało na wygodne budowanie i debugowanie[[12]](#footnote-12) projektu. W fazie implementacji korzystać będziemy w emulatora Shella EFI dostarczanego przez EDK II, dzięki temu możliwe będzie wcześniej wspomniane debugowanie aplikacji za pomocą VS.

Ostatnim etapem w procesie przygotowania środowiska było utworzenie katalogu na naszą aplikację[[13]](#footnote-13), umieszczenie go w jednym z modułów i dodanie wpisu w odpowiednim pliku .dsc[[14]](#footnote-14), tak aby został on uwzględniony podczas kompilacji. Folder, który utworzyłem to:

*EDK\_II\MdeModulePkg\Application\tetris*

Jak widać został on umieszczony w module Mde (Module Development Environment), a więc to w pliku

*EDK\_II\MdeModulePkg\MdeModulePkg.dsc*

należy dodać w sekcji [Components] wpis:

*MdeModulePkg/Application/tetris/tetris.inf*

Musimy również stworzyć plik .inf[[15]](#footnote-15) zawierający podstawowe informacje o naszej aplikacji.

[Defines]

INF\_VERSION = 0x00010005

BASE\_NAME = tetris

FILE\_GUID = 6987936E-ED34-44db-AE97-1FA5E4ED2116

MODULE\_TYPE = UEFI\_APPLICATION

VERSION\_STRING = 1.0

ENTRY\_POINT = UefiMain

[Sources]

main.c

[Packages]

MdePkg/MdePkg.dec

MdeModulePkg/MdeModulePkg.dec

[LibraryClasses]

UefiApplicationEntryPoint

UefiLib

Po wykonaniu powyższych czynności można przystąpić do próby napisania pierwszego programu w stylu "Hello World!" i odpalenia go w emulatorze, tak też uczyniłem.

Na następnej stronie przestawiony jest przykładowy kod[[16]](#footnote-16) wyświetlający w konsoli krótki tekst oraz efekt jego działania.

#include <Uefi.h>

#include <Library/UefiLib.h>

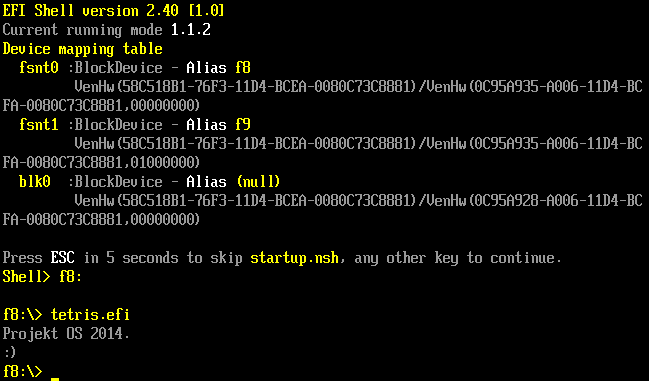
#include <Library/UefiApplicationEntryPoint.h>

EFI\_STATUS EFIAPI UefiMain( IN EFI\_HANDLE ImageHandle, IN EFI\_SYSTEM\_TABLE \*SystemTable ) {

Print( L"Projekt OS 2014.\n:)" );

return EFI\_SUCCESS;

}



W skład EDK II wchodzi wiele bibliotek[[17]](#footnote-17),[[18]](#footnote-18), bez których realizacja tego projektu byłaby niewykonalna, zapewniają one funkcjonalności podobne do standardowych bibliotek języka C. Do funkcji tych można zaliczyć:

* Alokację pamięci
* Obsługę klawiatury
* Obsługę wyświetlania
* Wsparcie dla debugowania
* Pomiar czasu

Funkcje te stanowią problematyczną część projektu i będą wymagały od nas zapoznania się ze specyfiką bibliotek zawierających je. Poza wyżej wymienionymi funkcjami, reszta kodu aplikacji nie będzie różniła się od "zwykłego" kodu w języku C, a ten mamy dobrze opanowany.

**Motywacja podjęcia tematu**

1. <http://pl.wikipedia.org/wiki/Tetris> [↑](#footnote-ref-1)
2. <http://tianocore.sourceforge.net/wiki/EDK_II_Overview> [↑](#footnote-ref-2)
3. <http://www.winiso.com/> [↑](#footnote-ref-3)
4. <https://www.virtualbox.org/wiki/Downloads> [↑](#footnote-ref-4)
5. <http://software.intel.com/en-us/articles/about-uefi> [↑](#footnote-ref-5)
6. <http://www.rodsbooks.com/efi-programming/why.html> [↑](#footnote-ref-6)
7. <https://svn.code.sf.net/p/edk2/code/trunk/edk2> [↑](#footnote-ref-7)
8. [http://tianocore.sourceforge.net/wiki/Windows\_systems#Build\_MdeModulePkg](http://tianocore.sourceforge.net/wiki/Windows_systems%23Build_MdeModulePkg) [↑](#footnote-ref-8)
9. <http://msdn.microsoft.com/en-us/library/1wea5zwe.aspx> [↑](#footnote-ref-9)
10. <http://msdn.microsoft.com/en-us/library/jj851030.aspx> [↑](#footnote-ref-10)
11. <http://uefi.blogspot.com/2013/06/how-to-set-up-edk2s-windows-hosted-uefi.html> [↑](#footnote-ref-11)
12. <http://sourceforge.net/projects/efidevkit/files/Edk%20Getting%20Started%20Guide%5B1%5D.0.41.pdf> rozdział 3.4 [↑](#footnote-ref-12)
13. [http://tianocore.sourceforge.net/wiki/Getting\_Started\_Writing\_Simple\_Application#5.29\_Create\_a\_project](http://tianocore.sourceforge.net/wiki/Getting_Started_Writing_Simple_Application%235.29_Create_a_project) [↑](#footnote-ref-13)
14. [http://tianocore.sourceforge.net/wiki/Getting\_Started\_Writing\_Simple\_Application#6.29\_Build\_your\_UEFI\_Application](http://tianocore.sourceforge.net/wiki/Getting_Started_Writing_Simple_Application%236.29_Build_your_UEFI_Application) [↑](#footnote-ref-14)
15. <http://tianocore.sourceforge.net/wiki/Getting_Started_Writing_MyHelloWorld.inf> [↑](#footnote-ref-15)
16. <http://tianocore.sourceforge.net/wiki/Getting_Started_Writing_MyHelloWorld.c> [↑](#footnote-ref-16)
17. <http://sourceforge.net/projects/efidevkit/files/Documents/Beginners%20handbook/EdkReferenceManual.pdf> [↑](#footnote-ref-17)
18. <http://sourceforge.net/projects/edk2/files/Specifications/History/MDE_Library_Spec.pdf> [↑](#footnote-ref-18)