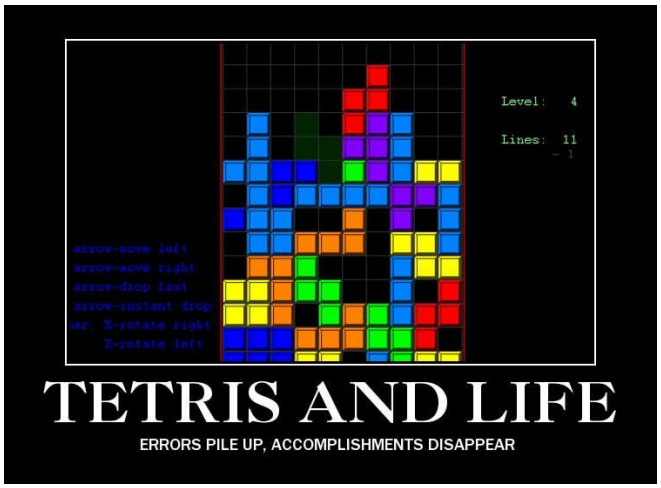
Krzysztof Łukasz Necel, 143301
Tomasz Gajger, 143218

Oprogramowanie Systemowe
Tetris w EFI

Tematyka pracy

Tematem pracy, realizowanej przez nas w ramach projektu z przedmiotu Oprogramowanie Systemowe, jest stworzenie gry podobnej do popularnego Tetrisa[1], ale działającej w środowisku shella EFI. Projekt będzie realizowany w oparciu o środowisko EDK II - EFI Developer Kit II[2].



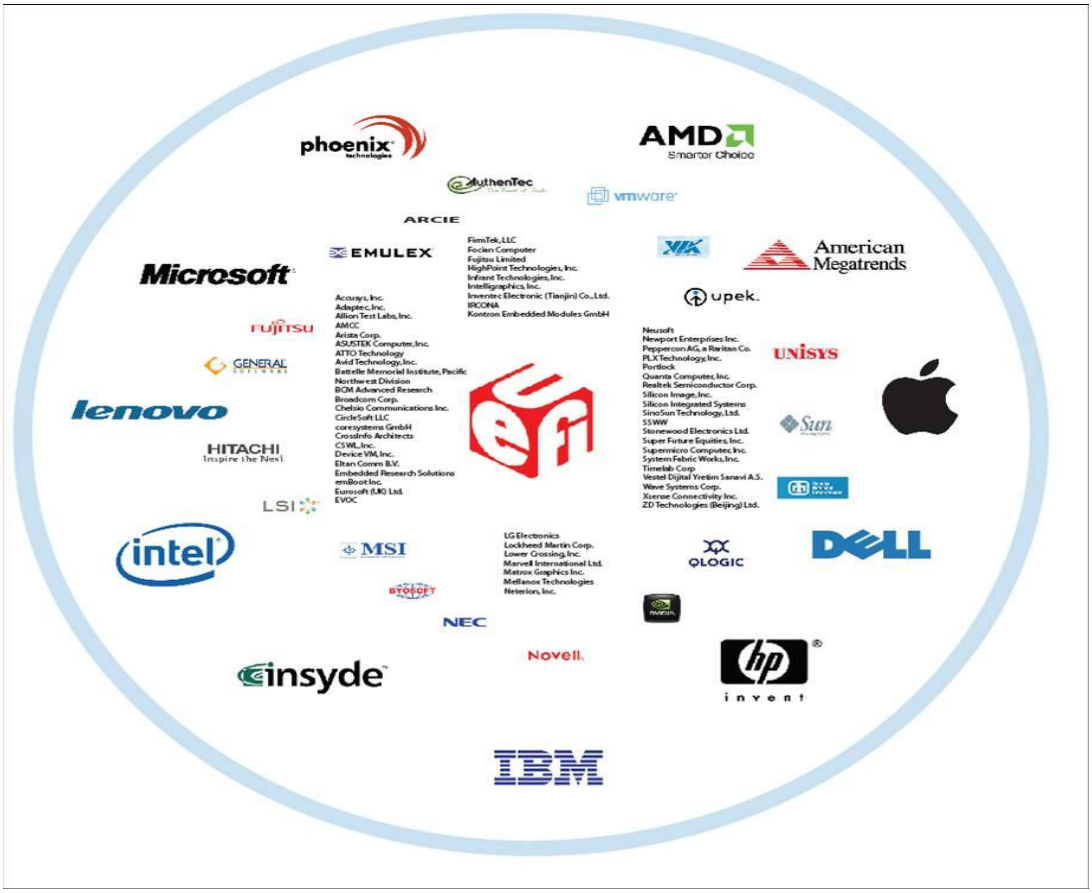
Celem, który chcemy osiągnąć, jest umieszczenie naszej gry w obrazie ISO[3], zamontowanie go w Virtual Boxie[4] i zaprezentowanie jej działania na maszynie wirtualnej.

[1] http://pl.wikipedia.org/wiki/Tetris
[2] http://tianocore.sourceforge.net/wiki/EDK_II_Overview
[3] http://www.winiso.com/
[4] https://www.virtualbox.org/wiki/Downloads

Przegląd technologii

Unified EFI Forum[5] 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.

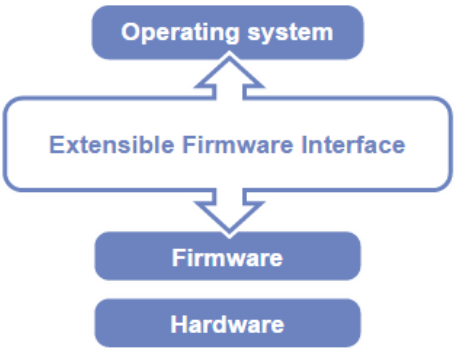


Często zadawane pytanie: Jaki jest związek pomiędzy EFI, a UEFI?

Odpowiedź: Specyfikacja UEFI bazuje 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.[6]

[5] http://www.uefi.org/
[6] http://software.intel.com/en-us/articles/about-uefi

Z powyższego paragrafu wynika, że popularna praktyka wykorzystywania pojęć UEFI i EFI zamiennie, nie stanowi poważnego błędu. Zmieniono jedynie nazewnictwo, ale ogólna idea stojąca za technologią pozostała niezmieniona, a samo UEFI jest, de facto, kontynuacją i rozwinięciem specyfikacji EFI.



UEFI jest specyfikacją definiującą interfejs programowy pomiędzy systemem operacyjnym, a firmware'em urządzeń.

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. 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.[7]

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 do załadowania jądra systemowego do pamięci i zainicjalizowania jego wykonania.

- **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.

- **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.

[7] http://www.rodsbooks.com/efi-programming/why.html

- **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.

Przykłady programów z powyższych kategorii można znaleźć z łatwością. 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.

Aplikacje shella UEFI vs. sterowniki UEFI.[8]

- Rdzeń UEFI dostarcza usługi i protokoły
- Sterowniki i aplikacje korzystają z usług UEFI

Sterowniki

- Posiadają wyższy priorytet
- Zwykle pozostają rezydentne

Aplikacje

- Napisane w celu wykonania określonego zadania
- Oczekuje się od nich, że po jego wykonaniu zakończą swoją pracę

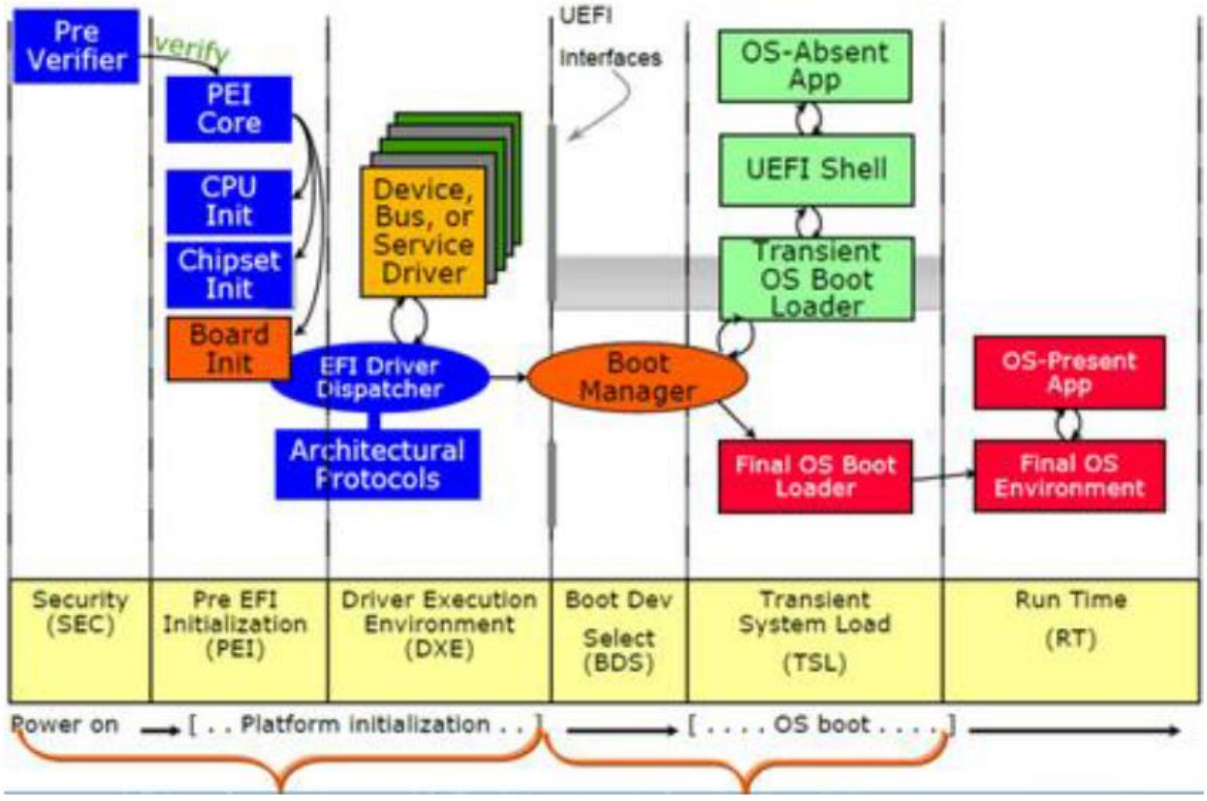
Zalety EFI.[9]

Interface	Legacy BIOS	UEFI
Architecture	x86 / X64 only	Agnostic
Mode	16 bit (real mode)	32/64 bit
Boot Partition	MBR (2.2 TB limit)	GPT (9.4 ZB* limit)
Runtime Services	No	Yes
Driver model	No	Yes
POST Graphics	VGA	Graphical Output Protocol (GOP)

[8] http://www.uefi.org/sites/default/files/resources/Insyde_Using_the_UEFI_Shell.pdf str. 9
[9] http://afis.ucc.ie/tbutler/BIOS%20and%20the%20UEFI.pdf, slajd 26

Start komputera wg EFI[10]

1. Ładowanie lekkiej warstwy PEI (Pre-EFI Initialization), która wykonuje większość rzeczy związanych z rozruchem komputera (=POST BIOS-u) jak np. inicjalizacja chipsetu, pamięci, enumeracja szyn danych itp.
2. Ładowanie DXE (Driver Execution Environment) dla zapewnienia podstawowego API dla sterowników.
3. Wykrywane są partycje dyskowe.
4. Uruchomienie programów rozruchowych OS-ów według kolejności z listy startowej.
5. Jeśli bootloader OS zadziała poprawnie, zostaną wyłączone usługi rozruchowe, a sterowanie zostanie przekazane do systemu operacyjnego.
6. Jeśli jednak nie ma znanego programu rozruchowego lub wszystkie okazały się błędne zostanie załadowana powłoka EFI Shell



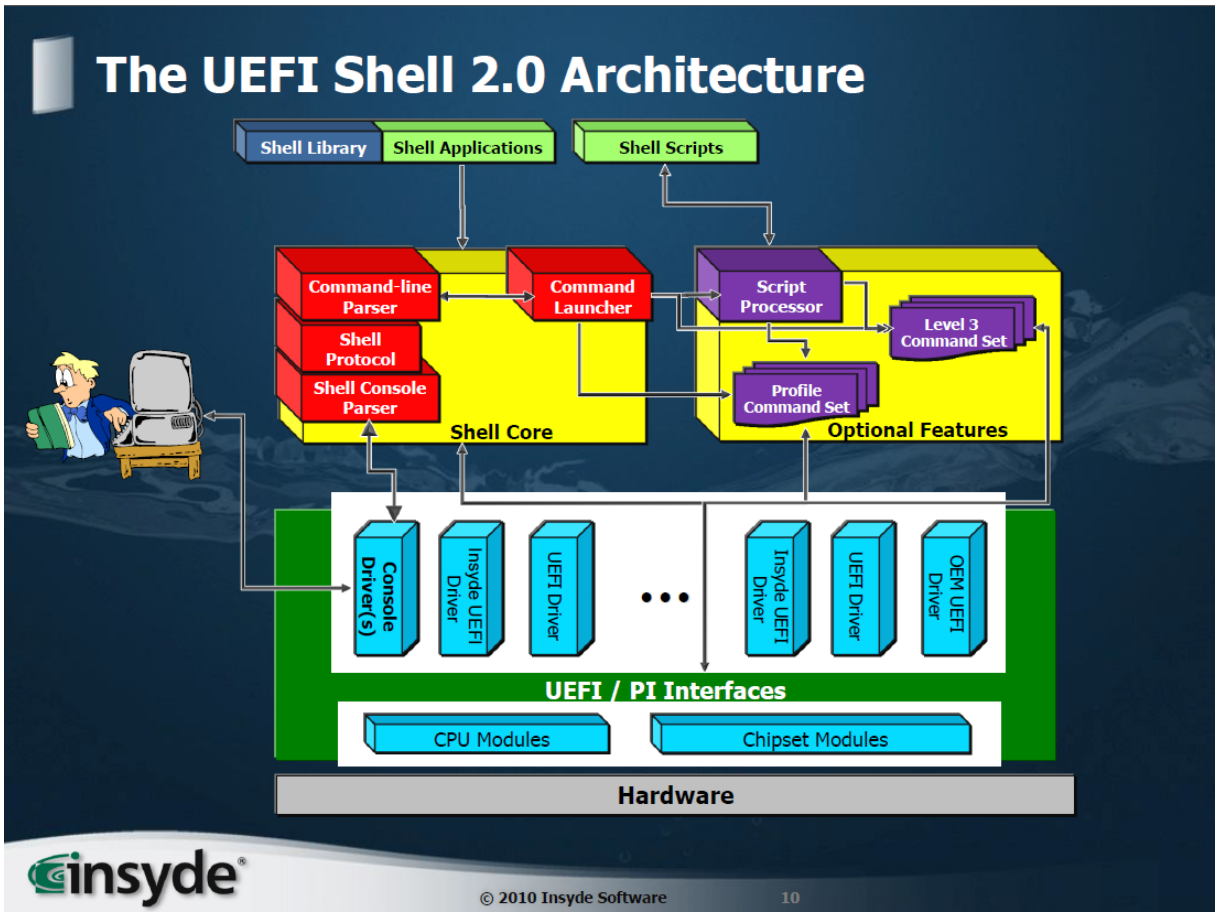
Rysunek przedstawiający schematycznie przebieg procesu bootowania[11]

[10] Wykład dr Dziubicha z Oprogramowania Systemowego: OS_2013_start_systemu.pdf, slajd 5

[11] http://afis.ucc.ie/tbutler/BIOS%20and%20the%20UEFI.pdf, slajd 19

Czym jest UEFI Shell 2.0?[12]

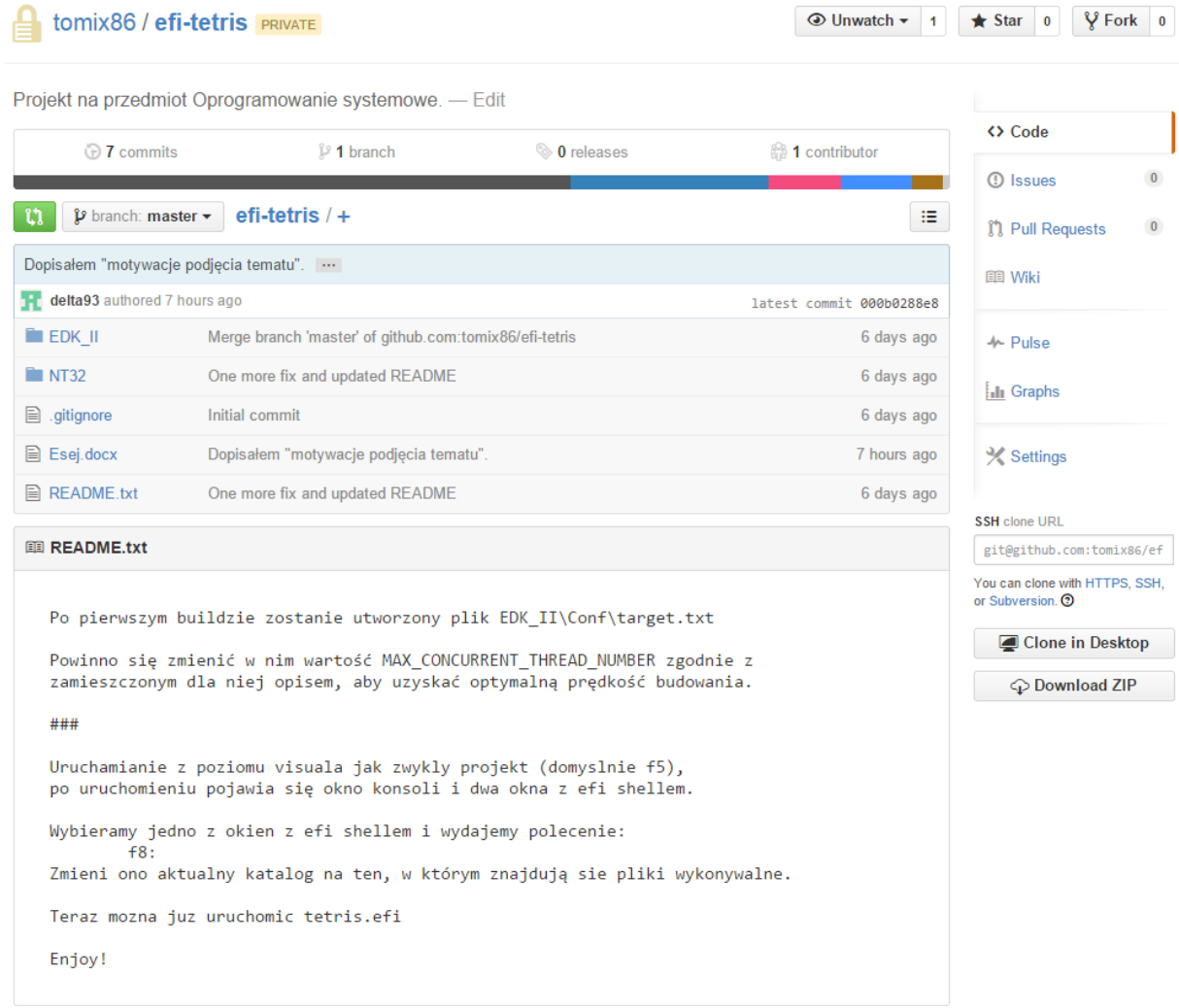
- Interaktywne rozszerzenie BIOSu
- Dostarcza środowisko uruchomieniowe dla programów
- Interpreter pozwalający na wykonanie plików skryptowych
- Bootowalny z urządzeń zewnętrznych
- Opcjonalnie dołączany do BIOSu jako urządzenie bootowalne.
- Wyglądem przypomina MS-DOS lub wiersz poleceń Linuxa
- Posiada pewien zbiór wbudowanych poleceń
    - Operacje na plikach
    - Zarządzanie sterownikami
    - Dostęp do urządzeń
    - Pobieranie informacji o stanie komputera
    - Dostęp do pamięci
    - Monitorowanie statusu BIOSu



[12] http://www.uefi.org/sites/default/files/resources/Insyde_Using_the_UEFI_Shell.pdf str. 8, 10 (w dokumencie tym opisana jest autorska implementacja Tiano wykorzystywana przez Insyde Software, nie będziemy z niej korzystać w projekcie, ale posłużymy się nią na potrzeby przeglądu technologii).

Przegląd środowiska

Wymianę plików prowadzić będziemy za pomocą GitHuba[13].



Wstępna konfiguracja środowiska polegała na pobraniu źródeł[14] EDK2 i próbie ich budowy z poziomu konsoli[15]. W tym miejscu wystąpiły drobne problemy, gdyż Visual Studio 2013[16] potraktowało pewne ostrzeżenia poziomu /W4 jako błędy, co uniemożliwiło kompilację. Musiałem więc dokonać ręcznej modyfikacji dwóch plików źródłowych, wyłączając ostrzeżenia[17,18] dyrektywą:

#pragma warning (disable: 4701 4703)

Po wykonaniu tego zabiegu kompilacja przebiegła pomyślnie.

---

[13] https://github.com/tomix86/efi-tetris (na chwilę obecną repozytorium jest prywatne)
[14] https://svn.code.sf.net/p/edk2/code/trunk/edk2
[15] http://tianocore.sourceforge.net/wiki/Windows_systems#Build_MdeModulePkg
[16] http://www.visualstudio.com/downloads/download-visual-studio-vs#d-express-windows-desktop
[17] http://msdn.microsoft.com/en-us/library/1wea5zwe.aspx
[18] http://msdn.microsoft.com/en-us/library/jj851030.aspx

Następnie skonfigurowałem[19] VS 2013, które wykorzystamy jako IDE, tak aby pozwalało na wygodne budowanie i debugowanie[20] projektu. W fazie implementacji korzystać będziemy z 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[21], umieszczenie go w jednym z modułów i dodanie wpisu w odpowiednim pliku .dsc[22], 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[23] 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.

---

[19] http://uefi.blogspot.com/2013/06/how-to-set-up-edk2s-windows-hosted-uefi.html
[20] http://sourceforge.net/projects/efidevkit/files/Edk%20Getting%20Started%20Guide%5B1%5D.0.41.pdf rozdział 3.4
[21] http://tianocore.sourceforge.net/wiki/Getting_Started_Writing_Simple_Application#5.29_Create_a_project
[22] http://tianocore.sourceforge.net/wiki/Getting_Started_Writing_Simple_Application#6.29_Build_your_UEFI_Application
[23] http://tianocore.sourceforge.net/wiki/Getting_Started_Writing_MyHelloWorld.inf

Poniżej przestawiony jest przykładowy kod[24] 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;
}
```

```
EFI Shell version 2.40 [1.0]
Current running mode 1.1.2
Device mapping table
  fsnt0 :BlockDevice - Alias f8
         VenHw(58C518B1-76F3-11D4-BCEA-0080C73C8881)/VenHw(0C95A935-A006-11D4-BC
FA-0080C73C8881,00000000)
  fsnt1 :BlockDevice - Alias f9
         VenHw(58C518B1-76F3-11D4-BCEA-0080C73C8881)/VenHw(0C95A935-A006-11D4-BC
FA-0080C73C8881,01000000)
  blk0  :BlockDevice - Alias (null)
         VenHw(58C518B1-76F3-11D4-BCEA-0080C73C8881)/VenHw(0C95A928-A006-11D4-BC
FA-0080C73C8881,00000000)

Press ESC in 5 seconds to skip startup.nsh, any other key to continue.
Shell> f8:

f8:\> tetris.efi
Projekt OS 2014.
:)
f8:\>
```

W skład EDK II wchodzi wiele bibliotek[25,26], 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.

[24] http://tianocore.sourceforge.net/wiki/Getting_Started_Writing_MyHelloWorld.c
[25] http://sourceforge.net/projects/efidevkit/files/Documents/Beginners%20handbook/EdkReferenceManual.pdf
[26] http://sourceforge.net/projects/edk2/files/Specifications/History/MDE_Library_Spec.pdf

Motywacja podjęcia tematu

Popularność środowiska UEFI w ostatnich latach dość drastycznie wzrosła i nadal rośnie. System BIOS został już niemal całkowicie wyparty przez UEFI w nowych komputerach klasy PC i nie tylko. Ilość aplikacji napisanych pod środowisko UEFI jest jednak nadal bardzo mała, tym bardziej jeśli weźmiemy pod uwagę fakt, iż środowisko to daje programistom stosunkowo duże pole do popisu. O tym jak duże, chcemy się przekonać podczas tworzenia naszego projektu i ustalić, w czym nas środowisko i dostępne narzędzia deweloperskie wspierają, a w czym ograniczają. Chcemy także nauczyć się podstaw tworzenia aplikacji pod UEFI oraz pokazać innym potencjał środowiska i to, że nie jest ono takie straszne jak wygląda na pierwszy rzut oka.