OPERATING SYSTEMS

UEFI GOP GRAPHICS LIBRARY

LAKSMAN SIVARAM SENTHILKUMAR
GAURAV CHAUHAN

WHAT IS UEFI

(U)EFI or (Unified) Extensible Firmware Interface is a specification for x86, x86-64, ARM, and Itanium platforms that defines a software interface between the operating system and the platform firmware/BIOS. The original EFI was developed in the mid-1990s by Intel for use developing firmware/BIOS for Itanium platforms.

In 2005 Intel transitioned the specification to a new working group called the Unified EFI Forum, consisting of companies such as AMD, Microsoft, Apple, and Intel itself. All modern PCs ship with UEFI firmware and UEFI is widely supported by both commercial and open source operating systems. Backwards compatibility is provided for legacy operating systems.

UEFI PROTOCOLS

The extensible nature of UEFI is built, to a large degree, around protocols. Protocols serve to enable communication between separately built modules, including drivers.

Drivers create protocols consisting of two parts. The body of a protocol is a C-style data structure known as a protocol interface structure, or just "interface". The interface typically contains an associated set of function pointers and data structures.

Every protocol has a GUID associated with it. The GUID serves as the name for the protocol.

UEFI BOOT AND RUNTIME SERVICES

EFI defines two types of services: Boot services and Runtime services. Boot services are available only while the firmware owns the platform, and they include text and graphical consoles on various devices, the bus, block and file services. Runtime services are still accessible while the operating system is running,they include services such as date, time and NVRAM access.

The Graphics Output Protocol provides runtime services. The operating system is permitted to directly write to the frame buffer provided by GOP during runtime mode, but it has to be initialized before exiting boot services since GOP is a boot time service.

UEFI GRAPHICS OUTPUT PROTOCOL

Graphics output is important in the pre-boot space to support modern firmware features. These features include the display of logos, the localization of output to any language, and setup and configuration screens. Graphics output may also be required as part of the startup of an operating system. There are potentially times in modern operating systems prior to the loading of a high performance OS graphics driver where access to graphics output device is required. The Graphics Output Protocol supports this capability by providing the EFI OS loader access to a hardware frame buffer and enough information to allow the OS to draw directly to the graphics output device.

OUR PROJECT

Our project is an abstraction layer on top of the GOP protocol which can be used like any other graphics library(SDL2, SFML, etc) but directly boots as an EFI executable. It can be used by small firmware tools and can be used to render images, rectangles, circles, lines and directly plot pixels.

The library also provides user input from keyboard and mouse pointer, and bitmap and vector font rendering.

LIBRARY FUNCTIONS

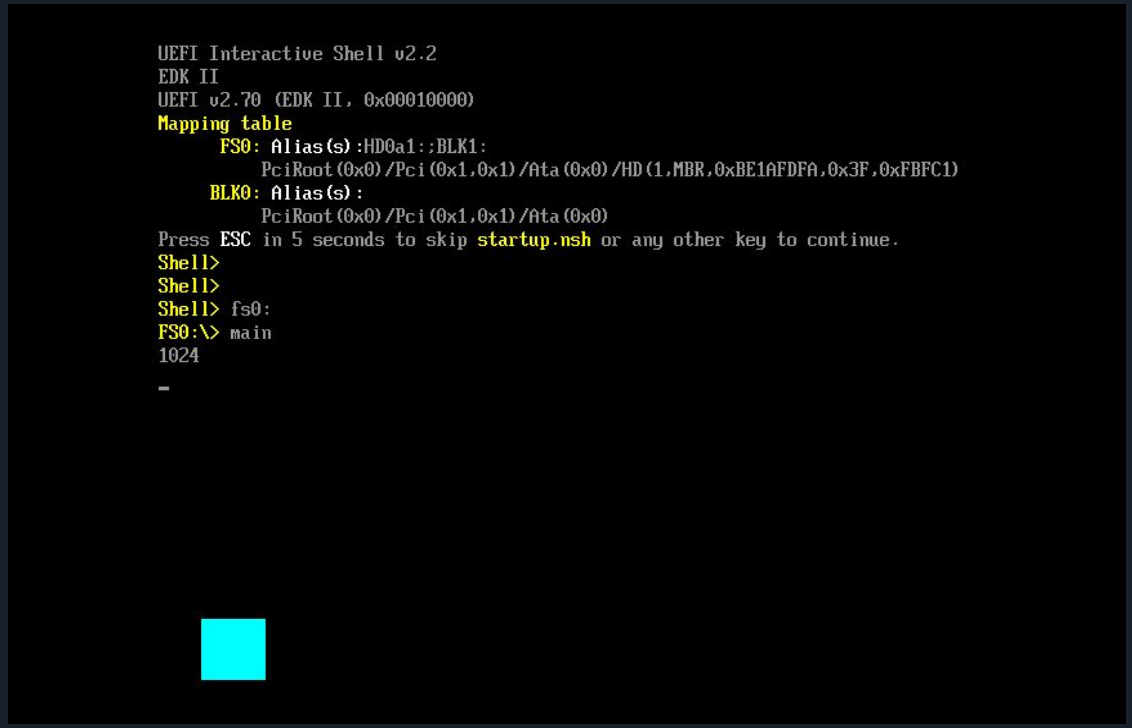
```
void clear_screen();
uint32_t *get_pixel(int x, int y);
void put_pixel(int x, int y, uint32_t pixel);
void draw_rect(int x, int y, int height, int width, int pixel);
void read_key(Key *key);
void set_timer(uint64_t trigger_time, efi_event_notify_t handler);
void ggl_init();
void render_text(const char *text,Font *font,int x, int y);
Image *load_image(const char *image);
void render_image(Image *image, int x, int y);
```

put_pixel IMPLEMENTATION

```
void put_pixel(int x, int y, uint32_t pixel)
{
   *((uint32_t *)(gop->Mode->FrameBufferBase + 4 *
gop->Mode->Information->PixelsPerScanLine * y + 4 * x)) = pixel;
}
```

We have to multiply PixelsPerScanLine by the number of bytes per pixel. That can be detected by examining the gop->Mode->Info->PixelFormat field.

SAMPLE PROGRAM



DEADLINE 1: JUNE 17

1. Add font rendering capabilities
2. Create a simple 2d game using our library
3. Implement multiple key input buffering

DEADLINE 2: SUMMER

1. Implement Image loading capabilities.
2. Port an existing game/NES emulator which uses SDL2 or create a game using our library.