

DoomUI

1

Кудинов Денис

О чём поговорим?

2

О современном и не очень

C

3D rendering

SwiftUI

Doom

План встречи

3

Основы рендеринга 3D

3D движок на SwiftUI

Собираем Doom под macOS

Собираем Doom под iOS

Запуск Doom на SwiftUI

План встречи

4

Основы рендеринга 3D

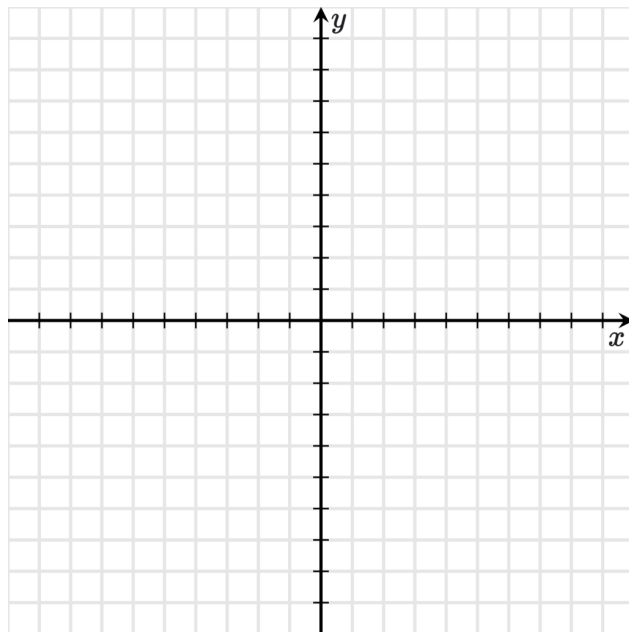
3D движок на SwiftUI

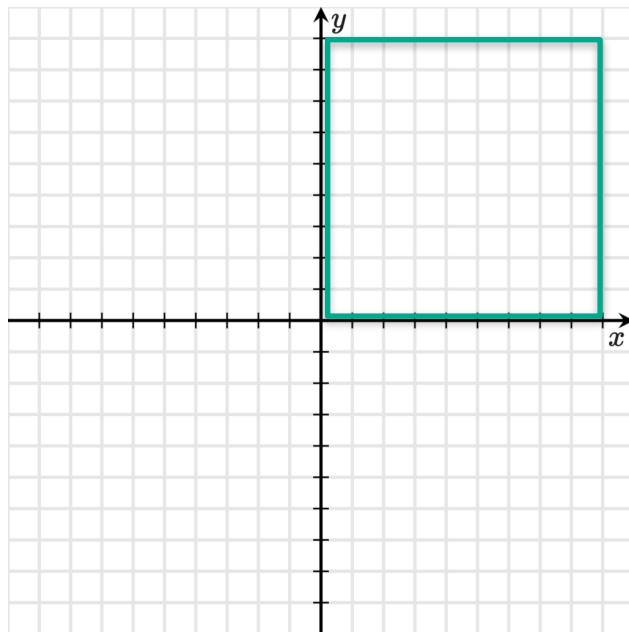
Собираем Doom под macOS

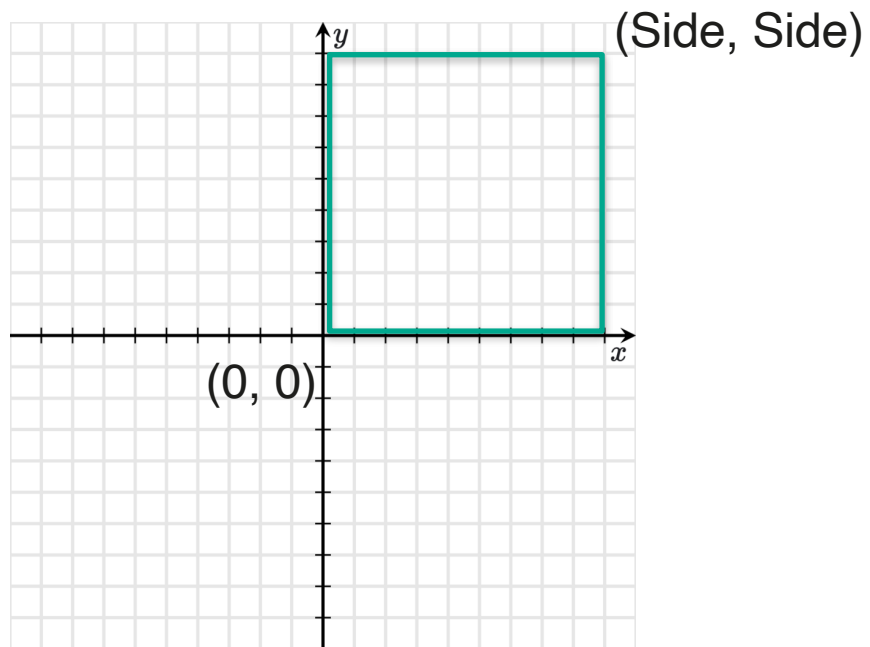
Собираем Doom под iOS

Запуск Doom на SwiftUI

Как на экране
рендерятся 2D
объекты?



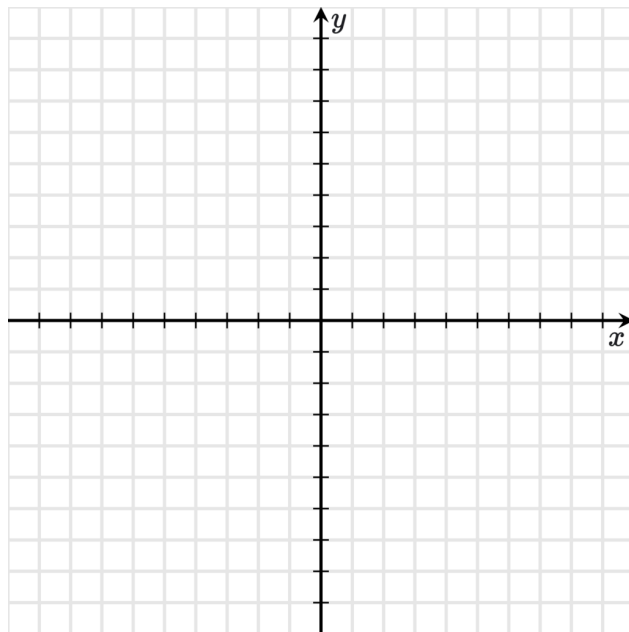


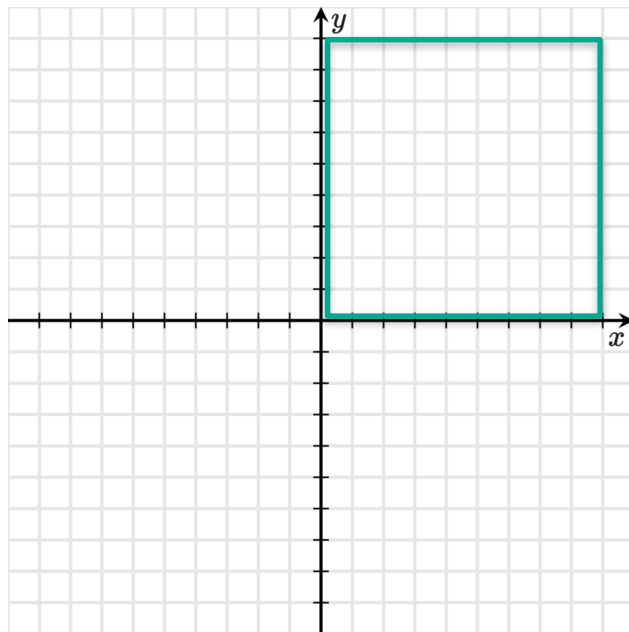


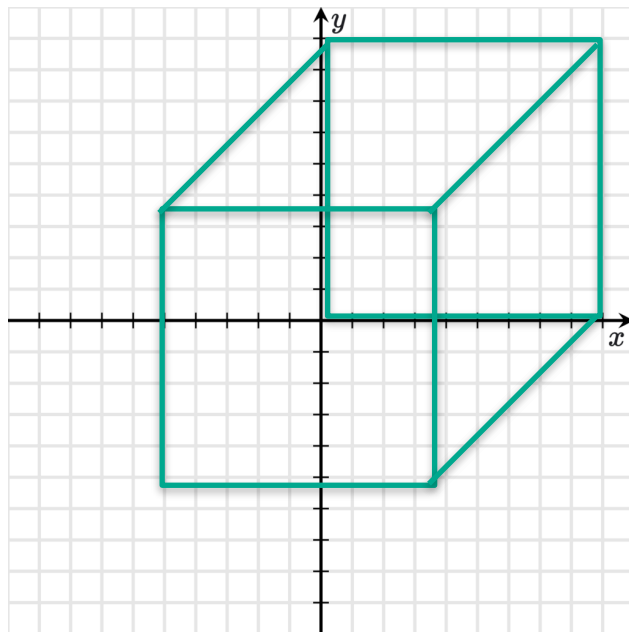
```
Rect(  
    Point(0, 0),  
    Point(side, side)  
)
```

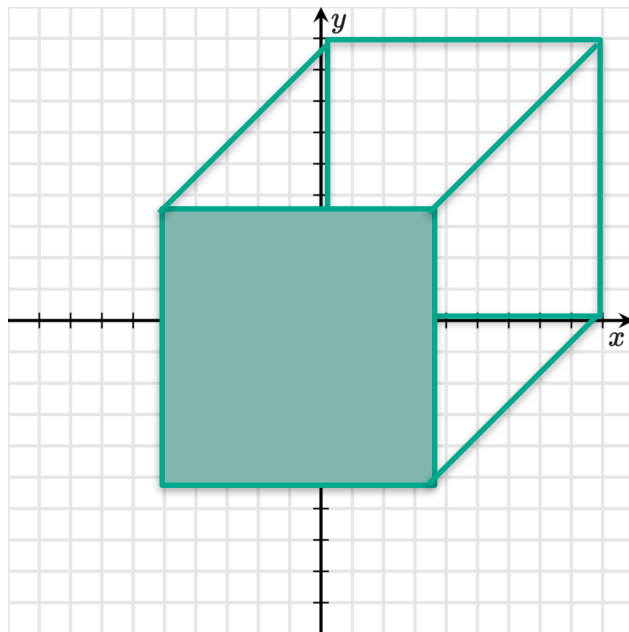
Как на экране
рендерятся 3D
объекты?

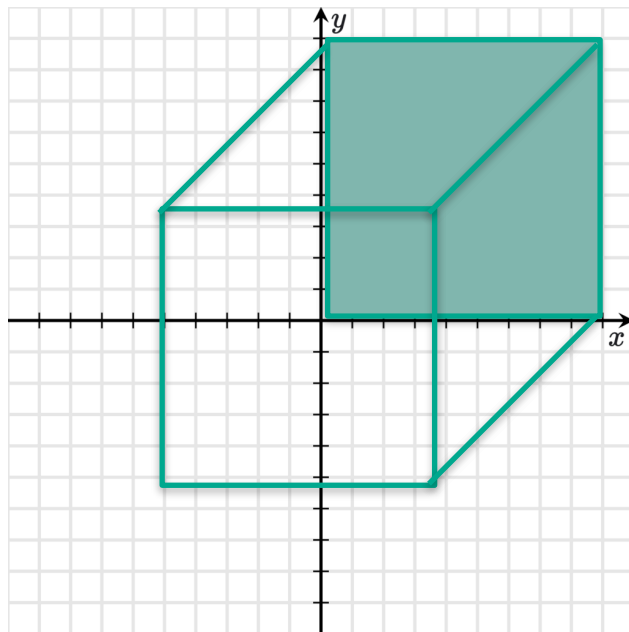


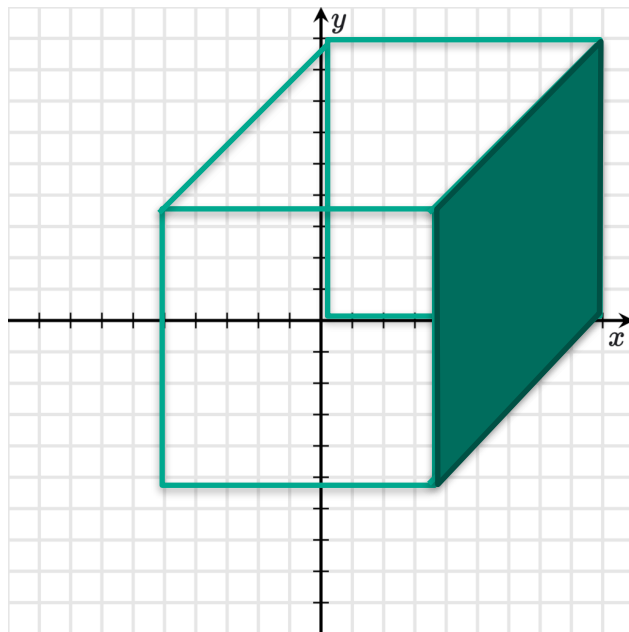




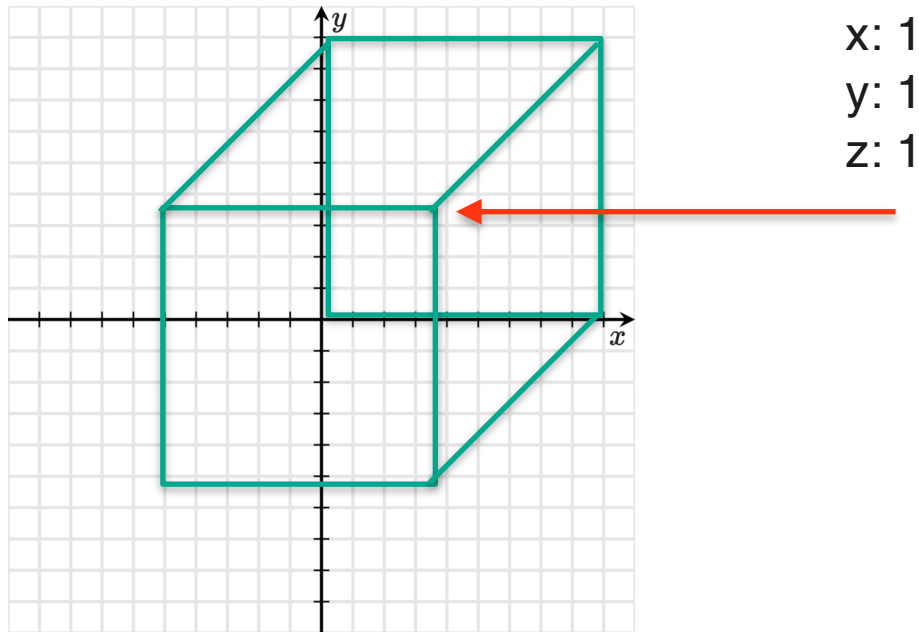








Нужно как-то научиться переводить точки из 3D в 2D проекции



Projection matrix

Позволяет провести
математические действия для
перевода из 3D в 2D

Относительно легко
программируется

Но об этом чуть позже

План встречи

20

Основы рендеринга 3D

3D движок на SwiftUI

Собираем Doom под macOS

Собираем Doom под iOS

Запуск Doom на SwiftUI

Кто такой “движок”?

умеет преобразовывать
3D → 2D

умеет отрисовывать картинку

в теории умеет обрабатывать
освещение, Ray tracing, ...

умеет вращать, приближать,
позиционировать элементы

Как будем рисовать на экране?

Как это обычно делают?

OpenGL

DirectX

Vulkan

Metal

SwiftUI Canvas

Path

Перфоманс низкий, но это
только POC

Можно рисовать треугольники
(да и вообще произвольные
формы)

SwiftUI так же работает в
связке с DisplayLink

CADisplayLink

Помогает рисовать картинку
frame by frame

Вызывает callback после
отрисовки кадра

Является чем-то средним
между таймером и обзёрвером
vsync

Можно рисовать на экране
сразу после очередного callback

CADisplayLink

Можно получать время между кадрами

Всегда для отрисовки лучше использовать его чем NSTimer

Можно ограничивать частоту кадров вручную

Projection matrix

$$\begin{bmatrix} \mathbf{d}_x \\ \mathbf{d}_y \\ \mathbf{d}_z \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta_x) & \sin(\theta_x) \\ 0 & -\sin(\theta_x) & \cos(\theta_x) \end{bmatrix} \begin{bmatrix} \cos(\theta_y) & 0 & -\sin(\theta_y) \\ 0 & 1 & 0 \\ \sin(\theta_y) & 0 & \cos(\theta_y) \end{bmatrix} \begin{bmatrix} \cos(\theta_z) & \sin(\theta_z) & 0 \\ -\sin(\theta_z) & \cos(\theta_z) & 0 \\ 0 & 0 & 1 \end{bmatrix} \left(\begin{bmatrix} \mathbf{a}_x \\ \mathbf{a}_y \\ \mathbf{a}_z \end{bmatrix} - \begin{bmatrix} \mathbf{c}_x \\ \mathbf{c}_y \\ \mathbf{c}_z \end{bmatrix} \right)$$

Projection matrix

$$\begin{bmatrix} \mathbf{d}_x \\ \mathbf{d}_y \\ \mathbf{d}_z \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta_x) & \sin(\theta_x) \\ 0 & -\sin(\theta_x) & \cos(\theta_x) \end{bmatrix} \begin{bmatrix} \cos(\theta_y) & 0 & -\sin(\theta_y) \\ 0 & 1 & 0 \\ \sin(\theta_y) & 0 & \cos(\theta_y) \end{bmatrix} \begin{bmatrix} \cos(\theta_z) & \sin(\theta_z) & 0 \\ -\sin(\theta_z) & \cos(\theta_z) & 0 \\ 0 & 0 & 1 \end{bmatrix} \left(\begin{bmatrix} \mathbf{a}_x \\ \mathbf{a}_y \\ \mathbf{a}_z \end{bmatrix} - \begin{bmatrix} \mathbf{c}_x \\ \mathbf{c}_y \\ \mathbf{c}_z \end{bmatrix} \right)$$

Вектор \mathbf{a} — позиция проецируемой точки

Projection matrix

$$\begin{bmatrix} \mathbf{d}_x \\ \mathbf{d}_y \\ \mathbf{d}_z \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta_x) & \sin(\theta_x) \\ 0 & -\sin(\theta_x) & \cos(\theta_x) \end{bmatrix} \begin{bmatrix} \cos(\theta_y) & 0 & -\sin(\theta_y) \\ 0 & 1 & 0 \\ \sin(\theta_y) & 0 & \cos(\theta_y) \end{bmatrix} \begin{bmatrix} \cos(\theta_z) & \sin(\theta_z) & 0 \\ -\sin(\theta_z) & \cos(\theta_z) & 0 \\ 0 & 0 & 1 \end{bmatrix} \left(\begin{bmatrix} \mathbf{a}_x \\ \mathbf{a}_y \\ \mathbf{a}_z \end{bmatrix} - \begin{bmatrix} \mathbf{c}_x \\ \mathbf{c}_y \\ \mathbf{c}_z \end{bmatrix} \right)$$

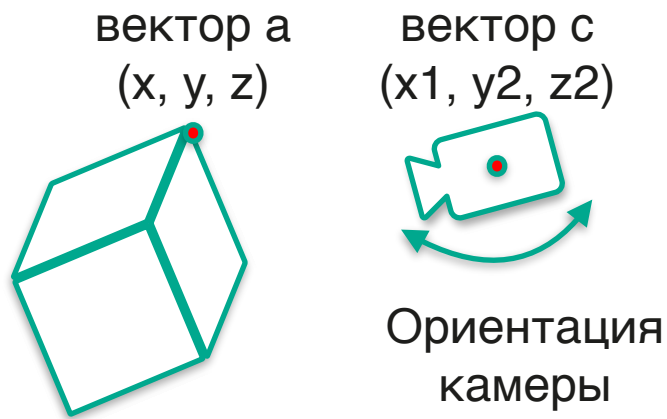
Вектор \mathbf{c} — позиция камеры

Projection matrix

$$\begin{bmatrix} \mathbf{d}_x \\ \mathbf{d}_y \\ \mathbf{d}_z \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta_x) & \sin(\theta_x) \\ 0 & -\sin(\theta_x) & \cos(\theta_x) \end{bmatrix} \begin{bmatrix} \cos(\theta_y) & 0 & -\sin(\theta_y) \\ 0 & 1 & 0 \\ \sin(\theta_y) & 0 & \cos(\theta_y) \end{bmatrix} \begin{bmatrix} \cos(\theta_z) & \sin(\theta_z) & 0 \\ -\sin(\theta_z) & \cos(\theta_z) & 0 \\ 0 & 0 & 1 \end{bmatrix} \left(\begin{bmatrix} \mathbf{a}_x \\ \mathbf{a}_y \\ \mathbf{a}_z \end{bmatrix} - \begin{bmatrix} \mathbf{c}_x \\ \mathbf{c}_y \\ \mathbf{c}_z \end{bmatrix} \right)$$

θ — ориентация камеры

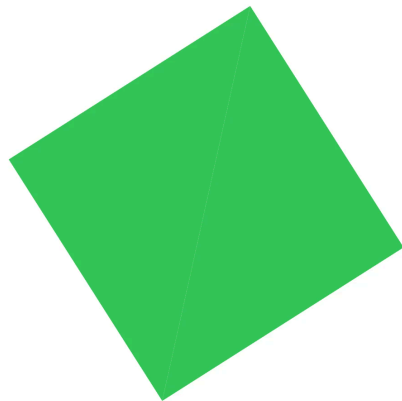
Projection matrix



```
var result = ...

for i in 0..
```

Движок “на минималках”



Основы рендеринга 3D

3D движок на SwiftUI

Собираем Doom под macOS

Собираем Doom под iOS

Запуск Doom на SwiftUI

Немного про Doom

Игра вышла в 1993 году
(30 лет назад)

Написана на C

*Компилируется на чём угодно

Факты про ИСХОДНИКИ

100% C

Доступны на Github

Были опубликованы в 1997
году

Только для Linux (урезанная
версия ресурсов)

Варианты сорцов

Оригинал на Github

[https://github.com/id-Software/
DOOM](https://github.com/id-Software/DOOM)

ChocolateDoom

Вариации портов дума под
другие платформы

Все варианты кроме
оригинала — не спортивно

Собираем
под macOS

Собираем под macOS

Открываем `.xcodeproj`

Нажимаем `cmd+b`

Собираем под macOS

Открываем .xcodeproj

Нажимаем cmd+b



Собираем под macOS

Makefile / make

Makefile — файл с инструкциями о том как собирать

make — утилита, которая запускает инструкции

```
$(0)/linuxxdoom: $(OBS) $(0)/i_main.o
    $(CC) $(CFLAGS) $(LDFLAGS) $(OBS) $(0)/i_main.o \
    -o $(0)/linuxxdoom $(LIBS)
```

```
$(0)/%.o: %.c
    $(CC) $(CFLAGS) -c $< -o $@
```

Собираем под macOS

Ищем makefile

Запускаем собранный
бинарник

Запускаем make



Запускаем
make

Запускаем make

```
$ make [11:32:40]
gcc -g -Wall -DNORMALUNIX -DLINUX -c doomstat.c -o linux/doomstat.o
In file included from doomstat.c:31:
In file included from ./doomstat.h:33:
In file included from ./doomdata.h:28:
./doomtype.h:42:10: fatal error: 'values.h' file not found
#include <values.h>
      ^~~~~~
1 error generated.
make: *** [linux/doomstat.o] Error 1
```

```
// Predefined with some OS.
#ifdef LINUX
#include <values.h>
#else
#define MAXCHAR          ((char)0x7f)
#define MAXSHORT         ((short)0x7fff)

// Max pos 32-bit int.
#define MAXINT            ((int)0x7fffffff)
#define MAXLONG           ((long)0x7fffffff)
#define MINCHAR          ((char)0x80)
#define MINSHORT         ((short)0x8000)

// Max negative 32-bit integer.
#define MININT            ((int)0x80000000)
#define MINLONG           ((long)0x80000000)
#endif
```

```
// Predefined with some OS.
#ifdef LINUX
#include <values.h>
#else
#define MAXCHAR          ((char)0x7f)
#define MAXSHORT         ((short)0x7fff)

// Max pos 32-bit int.
#define MAXINT           ((int)0xffffffff)
#define MAXLONG          ((long)0xffffffff)
#define MINCHAR          ((char)0x80)
#define MINSHORT         ((short)0x8000)

// Max negative 32-bit integer.
#define MININT           ((int)0x80000000)
#define MINLONG          ((long)0x80000000)
#endif
```

```
// Predefined with some OS.  
#ifdef LINUX  
#include <values.h>  
#else  
#define MAXCHAR          ((char)0x7f)  
#define MAXSHORT         ((short)0x7fff)  
  
// Max pos 32-bit int.  
#define MAXINT           ((int)0xffffffff)  
#define MAXLONG          ((long)0xffffffff)  
#define MINCHAR          ((char)0x80)  
#define MINSHORT         ((short)0x8000)  
  
// Max negative 32-bit integer.  
#define MININT           ((int)0x80000000)  
#define MINLONG          ((long)0x80000000)  
#endif
```

```
// Predefined with some OS.
#ifdef LINUX
#include <values.h>
#else
#define MAXCHAR          ((char)0x7f)
#define MAXSHORT         ((short)0x7fff)

// Max pos 32-bit int.
#define MAXINT           ((int)0x7fffffff)
#define MAXLONG          ((long)0x7fffffff)
#define MINCHAR          ((char)0x80)
#define MINSHORT         ((short)0x8000)

// Max negative 32-bit integer.
#define MININT           ((int)0x80000000)
#define MINLONG          ((long)0x80000000)
#endif
```

Кастомный дефайн MAX*, MIN*
Int, Char, Long, Short

values.h

Ещё и для 32-разрядных систем

values.h

```
/* This interface is obsolete.  New programs should use  
   <limits.h> and/or <float.h> instead of <values.h>.  */
```


MAXINT, ...

В macOS нет values.h

В его новых версиях (limits.h/
floats.h) нет таких дефайнов

```
// Predefined with some OS.
#ifdef LINUX
#include <values.h>
#else
#define MAXCHAR          ((char)0x7f)
#define MAXSHORT         ((short)0x7fff)

// Max pos 32-bit int.
#define MAXINT           ((int)0x7fffffff)
#define MAXLONG          ((long)0x7fffffff)
#define MINCHAR          ((char)0x80)
#define MINSHORT         ((short)0x8000)

// Max negative 32-bit integer.
#define MININT           ((int)0x80000000)
#define MINLONG          ((long)0x80000000)
#endif
```

Запускаем make

```
gcc -g -Wall -DNORMALUNIX -DLINUX -c i_sound.c -o linux/i_sound.o
i_sound.c:45:10: fatal error: 'linux/soundcard.h' file not found
#include <linux/soundcard.h>
          ^~~~~~
1 error generated.
```

```
//#include <linux/soundcard.h>
```

Запускаем make

```
gcc -g -Wall -DNORMALUNIX -DLINUX -c i_video.c -o linux/i_video.o
i_video.c:32:10: fatal error: 'X11/Xlib.h' file not found
#include <X11/Xlib.h>
           ^~~~~~
1 error generated.
```

X11

X Window System
Оконная система

Работает на UNIX-подобных
системах

Позволяет создавать окна для
отрисовки GUI

macOS не исключение

Устанавливаем X11

<https://support.apple.com/kb/DL641>

```
$ ln -s /opt/X11/include/X11 /usr/local/include/X11
```

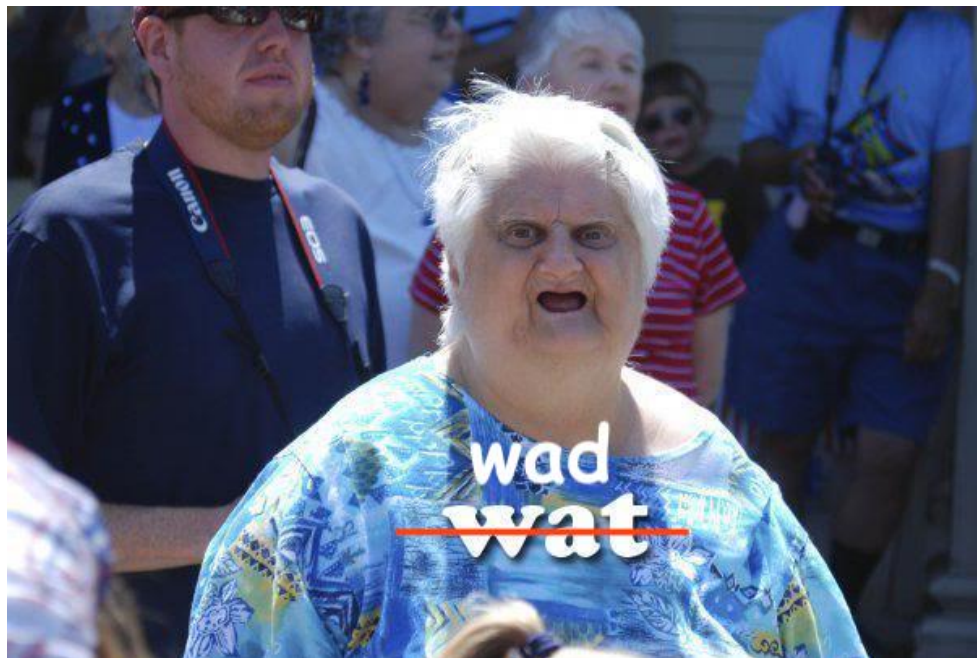
Запускаем make

```
m_misc.c:257:48: error: initializer element is not a compile-time constant  
    {"sndserver", (int *) &sndserver_filename, (int) "sndserver"},  
                                         ^~~~~~
```



```
Game mode indeterminate.  
                                Public DOOM - v1.10  
V_Init: allocate screens.  
M_LoadDefaults: Load system defaults.  
Z_Init: Init zone memory allocation daemon.  
W_Init: Init WADfiles.  
Error: W_InitFiles: no files found  
[1]    29617 segmentation fault  ./linuxxdoom
```

WAD



WAD файл

“Where's All the Data?”

Внутри wad файла всё, что связано с игрой

Содержит в себе директории и lump (куски)

Существуют версии
doom1.wad, doom2.wad, ...

WAD файл

Внутренности

План карты

Звуки

Текстуры

Все ресурсы в одном файле

└─o ./linuxxdoom

DOOM Shareware Startup v1.10

V_Init: allocate screens.

M_LoadDefaults: Load system defaults.

Z_Init: Init zone memory allocation daemon.

W_Init: Init WADfiles.

adding ./doom1.wad

=====

Shareware!

=====

M_Init: Init miscellaneous info.

R_Init: Init DOOM refresh daemon - [.[1] 31347 segmentation fault ./linuxxdoom

Debug

Искать по вхождениям
последних логов

???

Подключиться дебаггером

LLDB

Позволяет дебажить
приложение

Знаком iOS разработчикам

Позволяет ставить точки
останова

Отслеживание значений
локальных переменных

LLDB

Создать Target (какой бинарь
будем дебажить?)

Запустить процесс

Поставить точки останова

LLDB

```
↳ lldb  
(lldb) target create linuxxdoom
```

LLDB

```
(lldb) run
```

```
Process 31375 launched: 'D00M/linuxdoom-1.10/linux/linuxxdoom' (arm64)
```

LLDB

M_Init: Init miscellaneous info.

Process 31375 stopped

* thread #1, queue = 'com.apple.main-thread', stop reason = EXC_BAD_ACCESS (code=1, address=0x16fe104c8)

frame #0: 0x0000000100036650 linuxxdoom`R_InitTextures at r_data.c:542:21

```
539     {
540         patch->originx = SHORT(mpatch->originx);
541         patch->originy = SHORT(mpatch->originy);
-> 542         patch->patch = patchlookup[SHORT(mpatch->patch)];
543         if (patch->patch == -1)
544         {
545             I_Error ("R_InitTextures: Missing patch in texture %s",
```

Target 0: (linuxxdoom) stopped.

LLDB

M_Init: Init miscellaneous info.

Process 31375 stopped

* thread #1, queue = 'com.apple.main-thread', stop reason = EXC_BAD_ACCESS (code=1, address=0x16fe104c8)

frame #0: 0x0000000100036650 linuxxdoom`R_InitTextures at r_data.c:542:21

```
539     {
540         patch->originx = SHORT(mpatch->originx);
541         patch->originy = SHORT(mpatch->originy);
-> 542         patch->patch = patchlookup[SHORT(mpatch->patch)];
543         if (patch->patch == -1)
544         {
545             I_Error ("R_InitTextures: Missing patch in texture %s",
```

Target 0: (linuxxdoom) stopped.

LLDB

M_Init: Init miscellaneous info.

Process 31375 stopped

* thread #1, queue = 'com.apple.main-thread', stop reason = EXC_BAD_ACCESS (code=1, address=0x16fe104c8)

frame #0: 0x0000000100036650 linuxxdoom`R_InitTextures at r_data.c:542:21

```
539     {
540         patch->originx = SHORT(mpatch->originx);
541         patch->originy = SHORT(mpatch->originy);
-> 542         patch->patch = patchlookup[SHORT(mpatch->patch)];
543         if (patch->patch == -1)
544         {
545             I_Error ("R_InitTextures: Missing patch in texture %s",
```

Target 0: (linuxxdoom) stopped.

LLDB

```
M_Init: Init miscellaneous info.  
Process 31375 stopped  
* thread #1, queue = 'com.apple.main-thread', stop reason = EXC_BAD_ACCESS (code=1,  
address=0x16fe104c8)  
    frame #0: 0x0000000100036650 linuxxdoom`R_InitTextures at r_data.c:542:21  
    539     {  
    540         patch->originx = SHORT(mpatch->originx);  
    541         patch->originy = SHORT(mpatch->originy);  
-> 542         patch->patch = patchlookup[SHORT(mpatch->patch)];  
    543         if (patch->patch == -1)  
    544         {  
    545             I_Error ("R_InitTextures: Missing patch in texture %s",  
Target 0: (linuxxdoom) stopped.
```

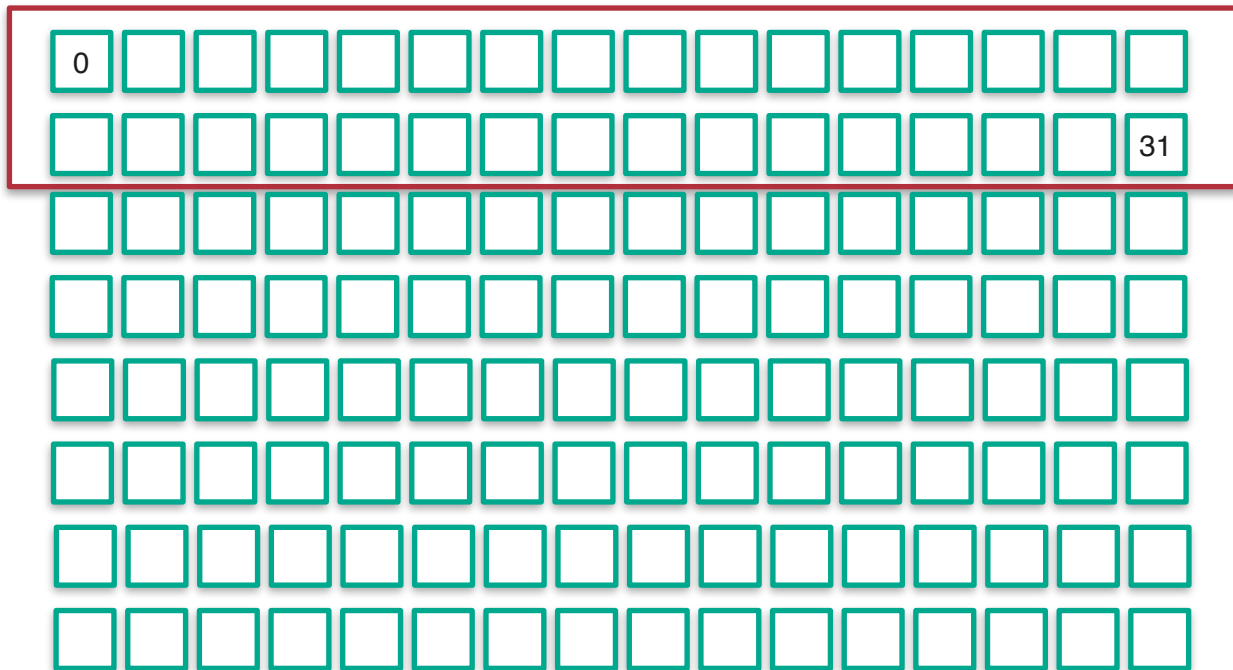
LLDB

```
M_Init: Init miscellaneous info.  
Process 31375 stopped  
* thread #1, queue = 'com.apple.main-thread', stop reason = EXC_BAD_ACCESS (code=1,  
address=0x16fe104c8)  
    frame #0: 0x0000000100036650 linuxxdoom`R_InitTextures at r_data.c:542:21  
    539     {  
    540         patch->originx = SHORT(mpatch->originx);  
    541         patch->originy = SHORT(mpatch->originy);  
-> 542         patch->patch = patchlookup[SHORT(mpatch->patch)];  
    543         if (patch->patch == -1)  
    544         {  
    545             I_Error ("R_InitTextures: Missing patch in texture %s",  
Target 0: (linuxxdoom) stopped.
```

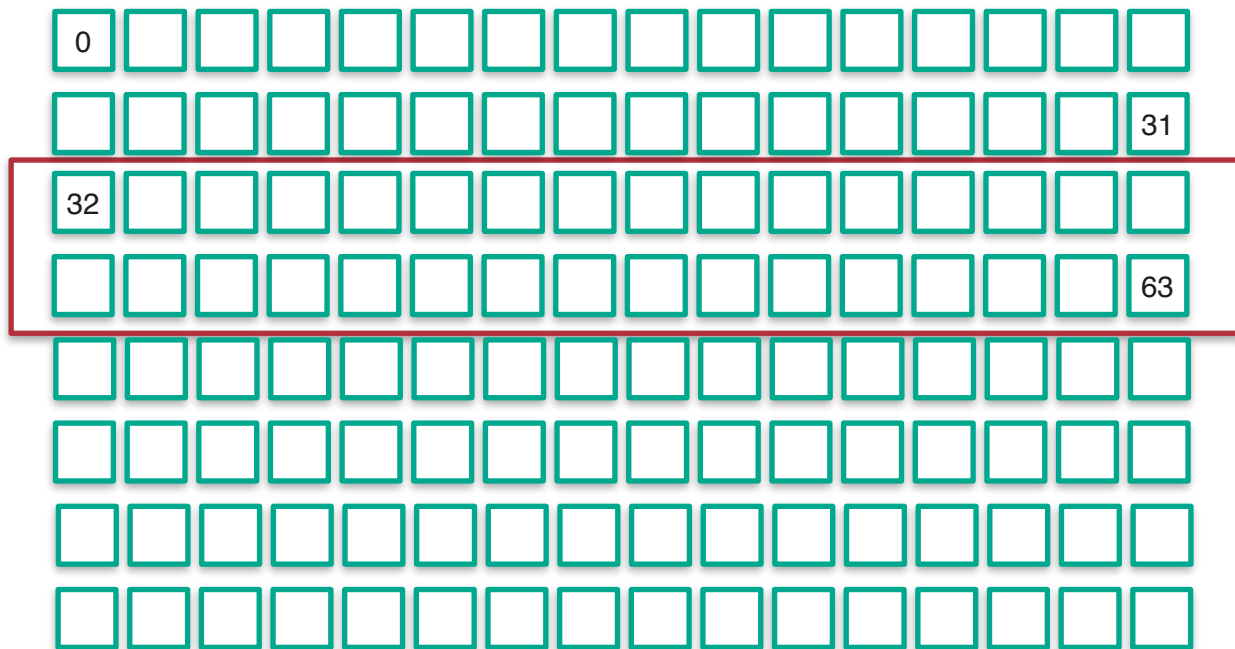
Debug

Подобных проблем было
довольно много

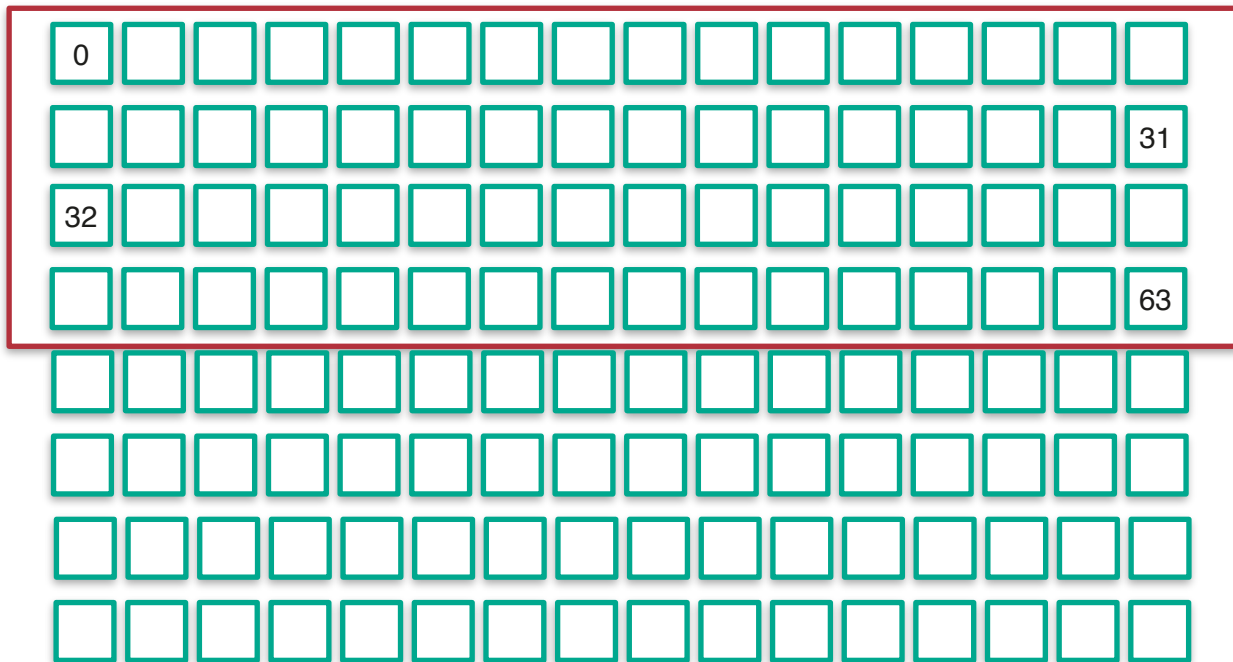
После изучения формата WAD
стало понятно что проблема в
размере указателей



32 бита



32 бита



64 бита

```
992     static fuck = 0;
994     // For debugging only
995     if (      fl->a.x < 0 || fl->a.x >= f_w
996         || fl->a.y < 0 || fl->a.y >= f_h
997         || fl->b.x < 0 || fl->b.x >= f_w
998         || fl->b.y < 0 || fl->b.y >= f_h)
999     {
1000         fprintf(stderr, "fuck %d \r", fuck++);
1001         return;
1002     }
1003
```



Основы рендеринга 3D

3D движок на SwiftUI

Собираем Doom под macOS

Собираем Doom под iOS

Запуск Doom на SwiftUI

Отличия iOS от macOS

Отсутствие кнопок для
управления

Отсутствует X11

Другая архитектура

Но есть кое-что получше

Разбираемся с рендерингом в X11

Что нужно для рендера в X11?

Формат фрейма может сильно варьироваться

Движок Doom формально “перекладывает байтики” в X11

В буфере находятся два фрейма: текущий и следующий

На вход X11 нужно получить массив с числами

Особенности рендеринга в Doom

Отрисовка в разрешении 320 x 200

Создание фрейма начинается с плана пола

Весь движок построен на концепции псевдо-3D

Стены формируются геометрически из текстур

Пишем свой рендер для Canvas

Разбиение пикселей на цвета

Отрисовка пикселей

Избавление от буфера (второго фрейма)

Достаём пиксели из массива для X11

```
for (size_t row = 0; row < ImageHeight; ++ row) {  
    for (size_t col = 0; col < ImageWidth; ++col) {  
        unsigned int data = image->data[row * ImageWidth + col];  
        int firstPixelNum = data & 0xff;  
        int secondPixelNum = data >> 16 & 0xff;  
  
        Pixel pixel;  
        pixel.r = colors[firstPixelNum].red;  
        pixel.g = colors[firstPixelNum].green;  
        pixel.b = colors[firstPixelNum].blue;  
        pixel.a = 255;  
  
        exportPixels->pixels[row * ImageWidth + col] = pixel;  
    }  
}
```

Достаём пиксели из массива для X11

```
for (size_t row = 0; row < ImageHeight; ++ row) {  
    for (size_t col = 0; col < ImageWidth; ++col) {  
        unsigned int data = image->data[row * ImageWidth + col];  
        int firstPixelNum = data & 0xff;  
        int secondPixelNum = data >> 16 & 0xff;  
  
        Pixel pixel;  
        pixel.r = colors[firstPixelNum].red;  
        pixel.g = colors[firstPixelNum].green;  
        pixel.b = colors[firstPixelNum].blue;  
        pixel.a = 255;  
  
        exportPixels->pixels[row * ImageWidth + col] = pixel;  
    }  
}
```

Достаём пиксели из массива для X11

```
for (size_t row = 0; row < ImageHeight; ++ row) {  
    for (size_t col = 0; col < ImageWidth; ++col) {  
        unsigned int data = image->data[row * ImageWidth + col];  
        int firstPixelNum = data & 0xff;  
        int secondPixelNum = data >> 16 & 0xff;  
  
        Pixel pixel;  
        pixel.r = colors[firstPixelNum].red;  
        pixel.g = colors[firstPixelNum].green;  
        pixel.b = colors[firstPixelNum].blue;  
        pixel.a = 255;  
  
        exportPixels->pixels[row * ImageWidth + col] = pixel;  
    }  
}
```

Что делать дальше?

Получили цвета и пиксели

Какой самый логичный способ
это сделать?

Нужно их как-то отрисовать в
iOS

Конечно же через SwiftUI

Canvas

Можно интегрировать в циклы
CADisplayLink

Позволяет рисовать любые
линии, а так же кастомные Path

В случае с самописным
движком “на треугольниках”

```
Canvas { context, size in
    for triangleIndex in triangleIndexes {
        let p1 = vertexArray[triangleIndex.0]
        let p2 = vertexArray[triangleIndex.1]
        let p3 = vertexArray[triangleIndex.2]

        var path = Path()
        path.move(to: CGPoint(x: Double(p1.x), y: Double(p1.y)))
        path.addLine(to: CGPoint(x: Double(p2.x), y: Double(p2.y)))
        path.addLine(to: CGPoint(x: Double(p3.x), y: Double(p3.y)))
        path.addLine(to: CGPoint(x: Double(p1.x), y: Double(p1.y)))

        context
            .fill(path, with: .color(colorProvider.update()))
    }
}
```

```
Canvas { context, size in
    for triangleIndex in triangleIndexes {
        let p1 = vertexArray[triangleIndex.0]
        let p2 = vertexArray[triangleIndex.1]
        let p3 = vertexArray[triangleIndex.2]

        var path = Path()
        path.move(to: CGPoint(x: Double(p1.x), y: Double(p1.y)))
        path.addLine(to: CGPoint(x: Double(p2.x), y: Double(p2.y)))
        path.addLine(to: CGPoint(x: Double(p3.x), y: Double(p3.y)))
        path.addLine(to: CGPoint(x: Double(p1.x), y: Double(p1.y)))

        context
            .fill(path, with: .color(colorProvider.update()))
    }
}
```

```
Canvas { context, size in
    for triangleIndex in triangleIndexes {
        let p1 = vertexArray[triangleIndex.0]
        let p2 = vertexArray[triangleIndex.1]
        let p3 = vertexArray[triangleIndex.2]

        var path = Path()
        path.move(to: CGPoint(x: Double(p1.x), y: Double(p1.y)))
        path.addLine(to: CGPoint(x: Double(p2.x), y: Double(p2.y)))
        path.addLine(to: CGPoint(x: Double(p3.x), y: Double(p3.y)))
        path.addLine(to: CGPoint(x: Double(p1.x), y: Double(p1.y)))

        context
            .fill(path, with: .color(colorProvider.update()))
    }
}
```

```
Canvas { context, size in
    for triangleIndex in triangleIndexes {
        let p1 = vertexArray[triangleIndex.0]
        let p2 = vertexArray[triangleIndex.1]
        let p3 = vertexArray[triangleIndex.2]

        var path = Path()
        path.move(to: CGPoint(x: Double(p1.x), y: Double(p1.y)))
        path.addLine(to: CGPoint(x: Double(p2.x), y: Double(p2.y)))
        path.addLine(to: CGPoint(x: Double(p3.x), y: Double(p3.y)))
        path.addLine(to: CGPoint(x: Double(p1.x), y: Double(p1.y)))

        context
            .fill(path, with: .color(colorProvider.update()))
    }
}
```

Основы рендеринга 3D

3D движок на SwiftUI

Собираем Doom под macOS

Собираем Doom под iOS

Запуск Doom на SwiftUI

Для начала нарисуем
простые пиксели

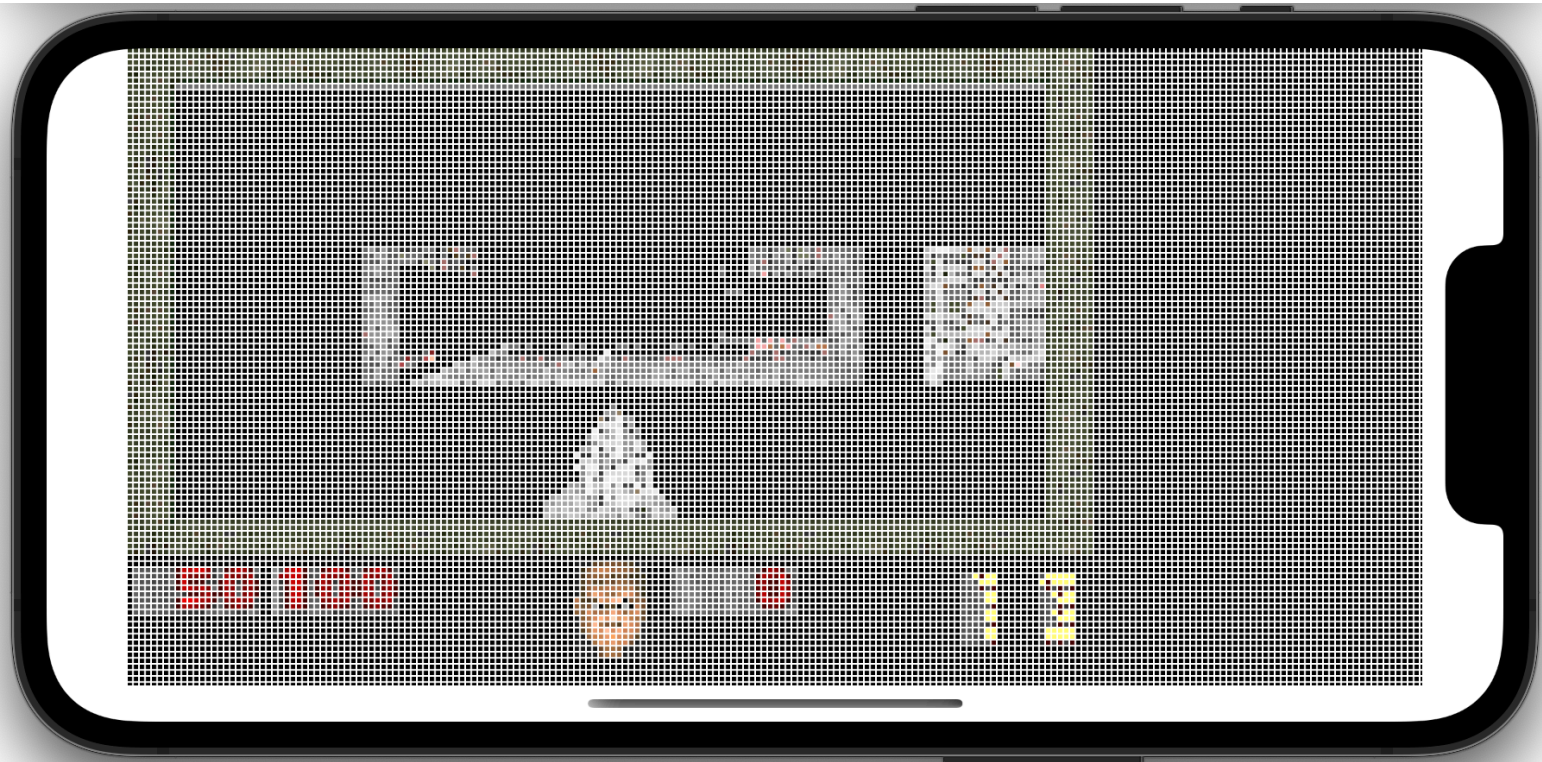
```
for pixelRow in pixels {  
    for pixel in pixelRow {  
        var squarePath = Path()  
        squarePath.move(to: CGPoint(x: x, y: y))  
        squarePath.addLine(to: CGPoint(x: x, y: y + size))  
        squarePath.addLine(to: CGPoint(x: x + size, y: y + size))  
        squarePath.addLine(to: CGPoint(x: x + size, y: y))  
        squarePath.addLine(to: CGPoint(x: x, y: y))  
        let color = UIColor(red: CGFloat(pixel.red) / 65535.0,  
                             green: CGFloat(pixel.green) / 65535.0,  
                             blue: CGFloat(pixel.blue) / 65535.0,  
                             alpha: CGFloat(pixel.alpha) / 255.0)  
        context.fill(squarePath, with: .color(Color(uiColor: color)))  
    }  
}
```

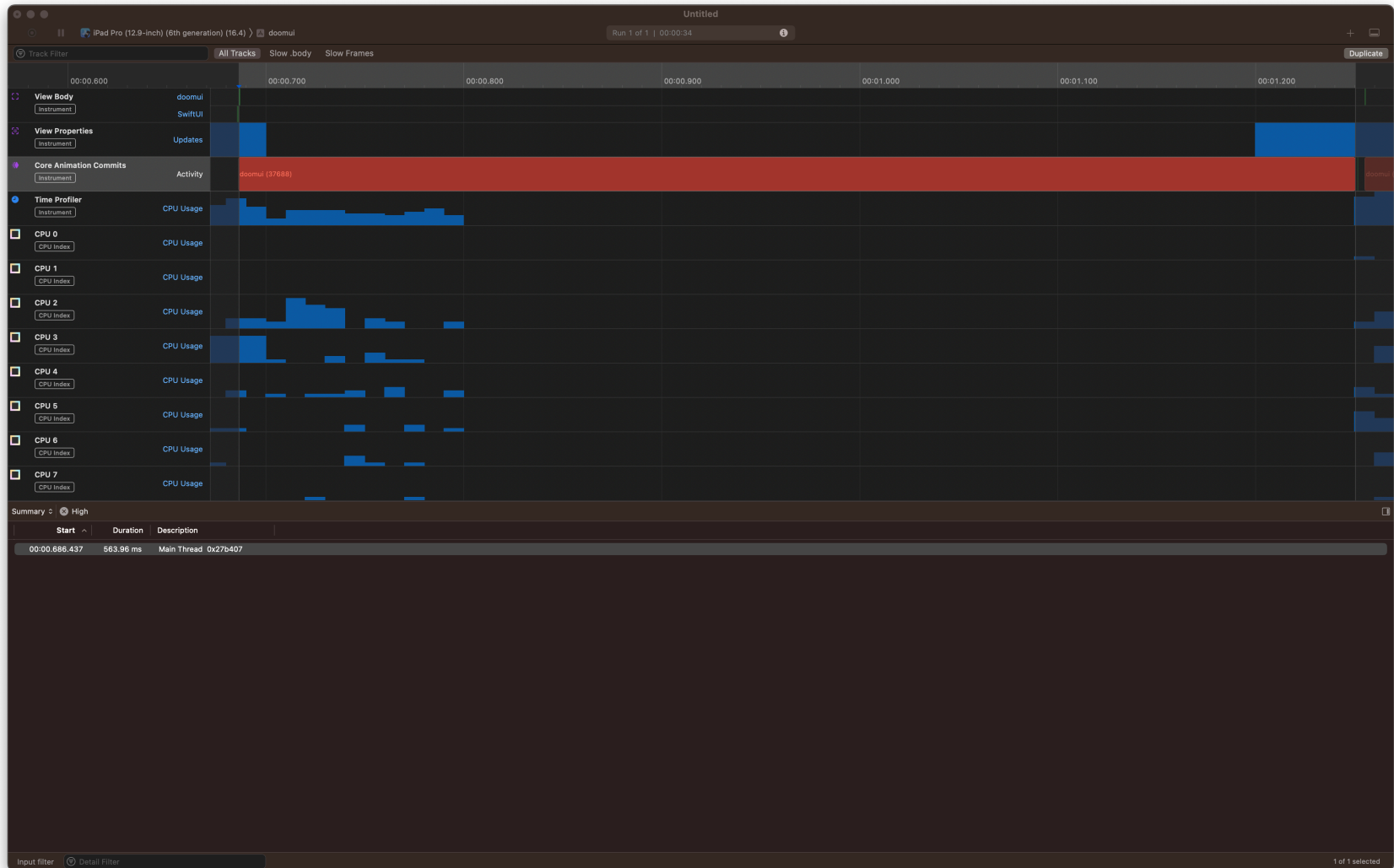


```
for pixelRow in pixels {  
    for pixel in pixelRow {  
        var squarePath = Path()  
        squarePath.move(to: CGPoint(x: x, y: y))  
        squarePath.addLine(to: CGPoint(x: x, y: y + size))  
        squarePath.addLine(to: CGPoint(x: x + size, y: y + size))  
        squarePath.addLine(to: CGPoint(x: x + size, y: y))  
        squarePath.addLine(to: CGPoint(x: x, y: y))  
        let color = UIColor(red: CGFloat(pixel.red) / 65535.0,  
                             green: CGFloat(pixel.green) / 65535.0,  
                             blue: CGFloat(pixel.blue) / 65535.0,  
                             alpha: CGFloat(pixel.alpha) / 255.0)  
        context.fill(squarePath, with: .color(Color(uiColor: color)))  
    }  
}
```

```
for pixelRow in pixels {  
    for pixel in pixelRow {  
        var squarePath = Path()  
        squarePath.move(to: CGPoint(x: x, y: y))  
        squarePath.addLine(to: CGPoint(x: x, y: y + size))  
        squarePath.addLine(to: CGPoint(x: x + size, y: y + size))  
        squarePath.addLine(to: CGPoint(x: x + size, y: y))  
        squarePath.addLine(to: CGPoint(x: x, y: y))  
        let color = UIColor(red: CGFloat(pixel.red) / 65535.0,  
                             green: CGFloat(pixel.green) / 65535.0,  
                             blue: CGFloat(pixel.blue) / 65535.0,  
                             alpha: CGFloat(pixel.alpha) / 255.0)  
        context.fill(squarePath, with: .color(Color(uiColor: color)))  
    }  
}
```







Какие-нибудь выводы?

106

SwiftUI не совсем готов к
такому повороту

2 FPS тоже FPS

Большую часть времени
отъедает отрисовка, а не
внутренняя кухня SwiftUI

Можно уменьшать количество пикселей, и
рисовать “через фрейм” для увеличения
перформанса

А теперь что-то
поинтереснее

Как насчёт Toggle (UISwitch)
вместо пикселей?

Что потребуется?

Создать $W * H$ свичей

Применить к ним цвета (они же всё-таки пиксели)

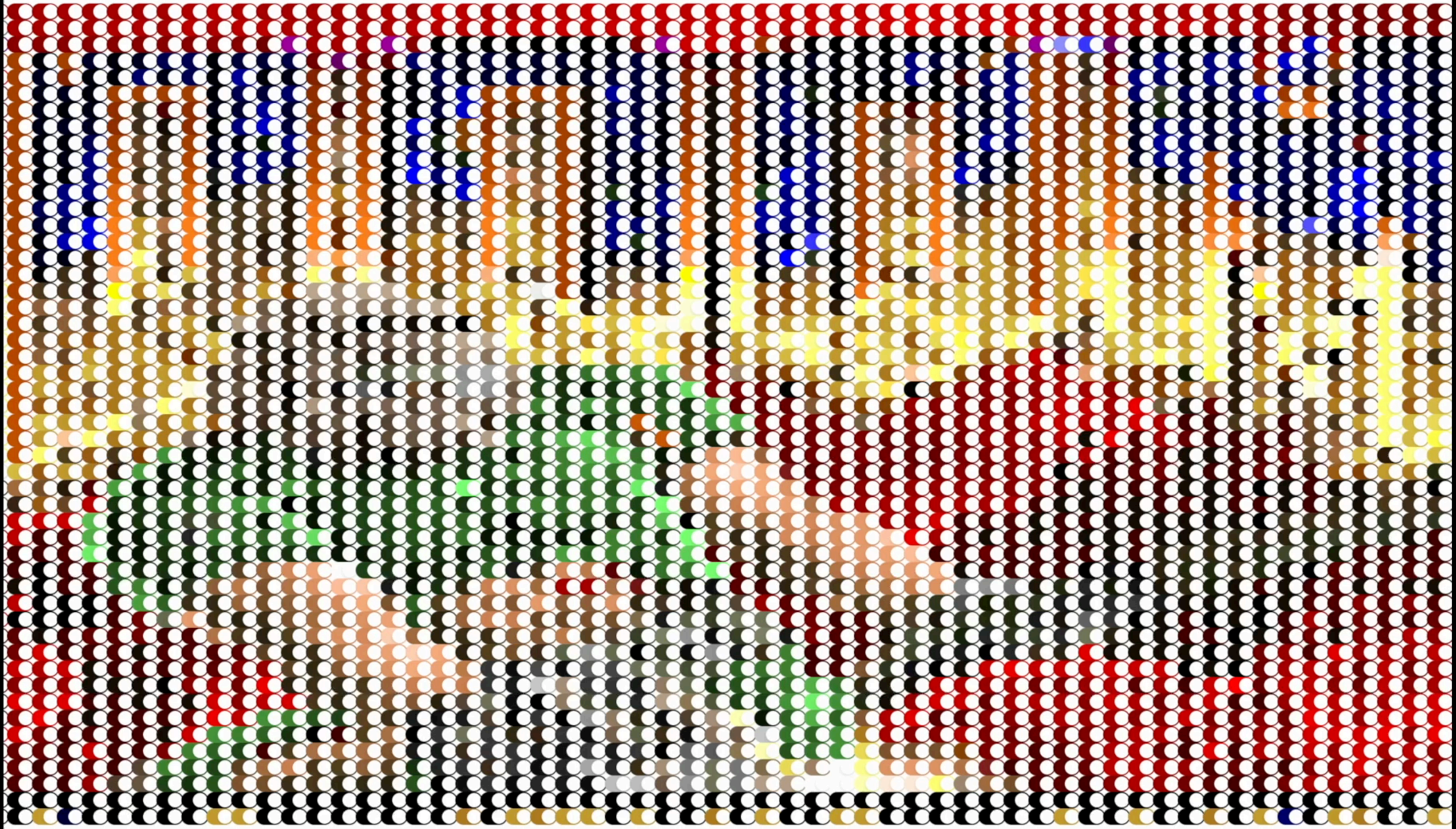
Научиться их распределять по экрану

```
VStack(spacing: 0) {  
  ForEach(pixels) { pixelRow in  
    HStack(spacing: -10) {  
      ForEach(pixelRow.pixels) { pixel in  
        let color = ...  
        Toggle("", isOn: $isOn)  
          .toggleStyle(SwitchToggleStyle(tint: color))  
      }  
    }  
  }  
}
```

```
VStack(spacing: 0) {  
  ForEach(pixels) { pixelRow in  
    HStack(spacing: -10) {  
      ForEach(pixelRow.pixels) { pixel in  
        let color = ...  
        Toggle("", isOn: $isOn)  
          .toggleStyle(SwitchToggleStyle(tint: color))  
      }  
    }  
  }  
}
```

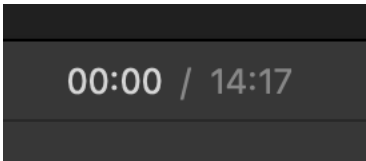
```
VStack(spacing: 0) {  
  ForEach(pixels) { pixelRow in  
    HStack(spacing: -10) {  
      ForEach(pixelRow.pixels) { pixel in  
        let color = ...  
        Toggle("", isOn: $isOn)  
          .toggleStyle(SwitchToggleStyle(tint: color))  
      }  
    }  
  }  
}
```

```
let color = Color(uiColor: UIColor(red: CGFloat(pixel.pixel.red) / 65535.0,  
                                     green: CGFloat(pixel.pixel.green) / 65535.0,  
                                     blue: CGFloat(pixel.pixel.blue) / 65535.0,  
                                     alpha: CGFloat(pixel.pixel.alpha) / 255.0))
```



Чего так
медленно?

Ну, во-первых,
это ещё быстро



00:00 / 14:17

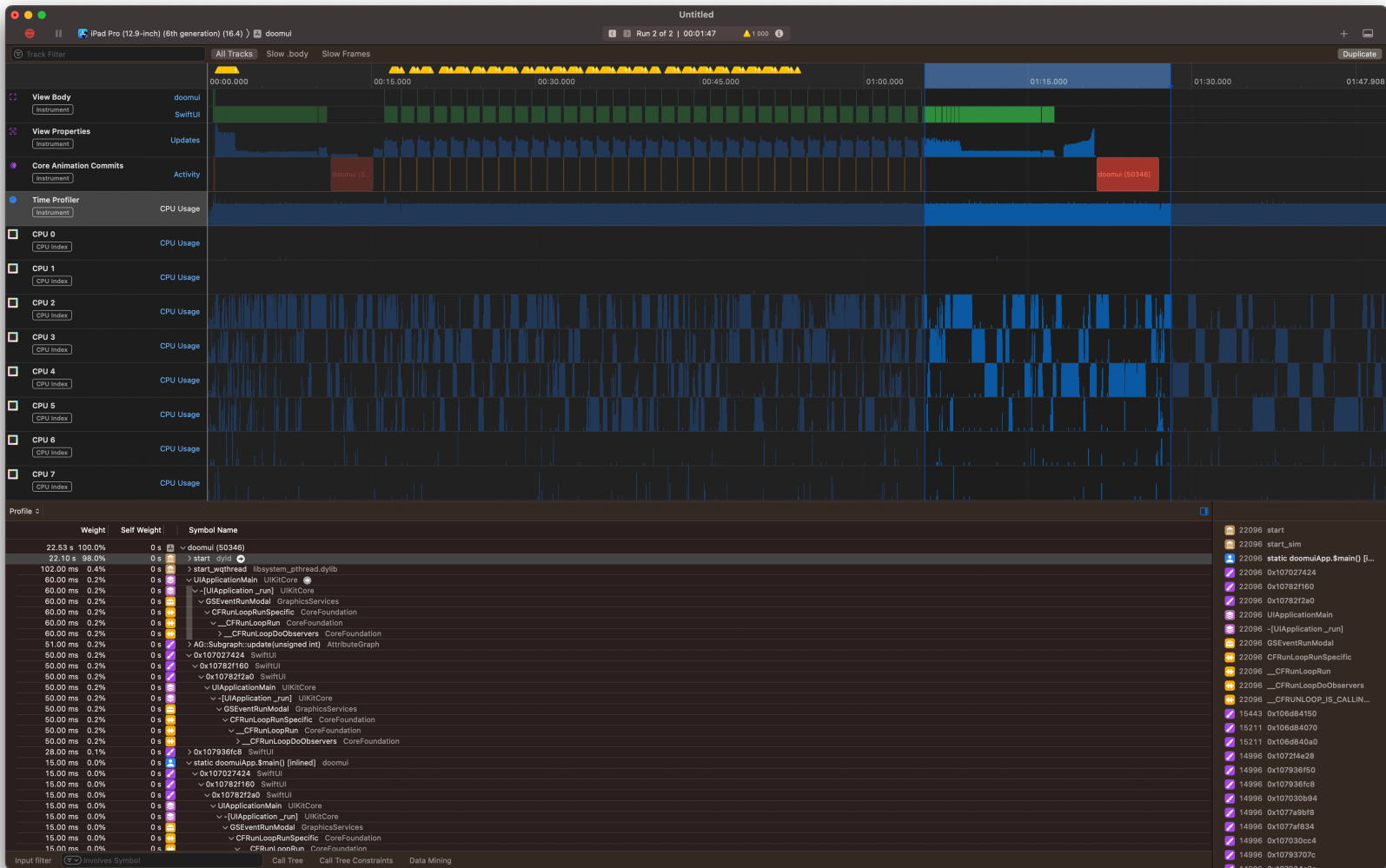
Первый фрейм
рендерится “быстро”

```
DrawCallback() frame # 1  
Start rendering  
Frame rendered in: 13.919343948364258
```

Остальные
значительно дольше

```
DrawCallback() frame # 43  
Start rendering  
Frame rendered in: 51.26048803329468
```

Заглянем в Profiler



20.19 s	76.2%	0 s		0x107397070 SwiftUI
25.29 s	78.2%	0 s		0x107934a3c SwiftUI
25.02 s	77.4%	110.00 ms		AG::Subgraph::update(unsigned int) AttributeGraph
22.08 s	68.3%	7.00 ms		AG::Graph::update_attribute(AG::data::ptr<AG::Node>, unsigned int) AttributeGraph
22.06 s	68.2%	193.00 ms		> AG::Graph::UpdateStack::update() AttributeGraph
9.00 ms	0.0%	9.00 ms		pthread_setspecific libsystem_thread.dylib
2.00 ms	0.0%	2.00 ms		pthread_getspecific libsystem_thread.dylib
1.00 ms	0.0%	1.00 ms		DYLD-STUB\$\$pthread_getspecific AttributeGraph
1.00 ms	0.0%	1.00 ms		0x106b9db80 SwiftUI
2.80 s	8.6%	0 s		AG::Graph::invalidate_subgraphs() AttributeGraph
10.00 ms	0.0%	10.00 ms		CFRetain CoreFoundation
10.00 ms	0.0%	0 s		> std::_1::stack<util::cf_ptr<AGSubgraphStorage*>, AG::vector<util::cf_ptr<AGSubgraphStorage*>, 32ul, unsigned long>>::pop[abi:v15006]() AttributeGraph
4.00 ms	0.0%	4.00 ms		_CFRetain CoreFoundation
3.00 ms	0.0%	3.00 ms		_CFRelease CoreFoundation
1.00 ms	0.0%	1.00 ms		DYLD-STUB\$\$CFRelease AttributeGraph
1.00 ms	0.0%	1.00 ms		CFRelease CoreFoundation
1.00 ms	0.0%	1.00 ms		AG::Graph::UpdateStack::update() AttributeGraph
204.00 ms	0.6%	0 s		> 0x107a353f0 SwiftUI
56.00 ms	0.1%	0 s		> 0x1072f4e28 SwiftUI

Что говорит Profiler?

В этот раз Core Animation уже кажется не виноват

Работой с графом занимается фреймворк AttributedGraph (AG)

Основное время уходит на работу с графом View SwiftUI

Перфоманс супер низкий на больших объёмах View

Кнопки?

Подход тот же что и в Toggle

Задаём положение и цвета

Создаём *Stack с кнопками

Попробуем 320* 200?

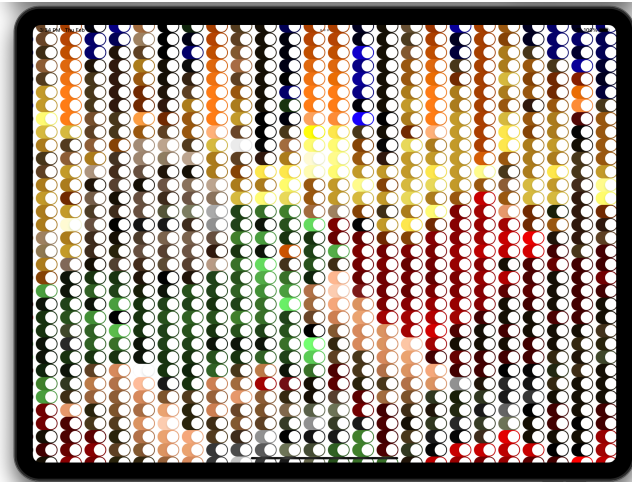
125

[MTLTextureDescriptorInternal validateWithDevice:],
line 1344: error 'Texture Descriptor Validation

MTLTextureDescriptor has height (8200) greater than the maximum allowed size of 8192

```
VStack(spacing: 0) {  
  ForEach(pixels) { pixelRow in  
    HStack(spacing: 0) {  
      ForEach(pixelRow.pixels) { pixel in  
        let color = Color(...)  
        Button("B") {  
  
          }  
          .foregroundColor(color)  
          .background(color)  
        }  
      }  
    }  
  }  
}
```





Ну вот и всё на сегодня :)