

Компьютерная графика

Лекция 1: Введение в курс

2023

Лекции:

- Лисица Никита Игоревич (Яндекс)
- lisyarus@gmail.com
- +7(952)276-70-50

Практики:

- Садовский Владимир Сергеевич (NauEngine)
- vovacolt@yandex.ru
- +7(988)581-87-78

Как устроен курс

github.com/lisyarus/graphics-course-slides/pdf

github.com/lisyarus/graphics-course-practice

Как устроен курс

github.com/lisyarus/graphics-course-slides/pdf

github.com/lisyarus/graphics-course-practice

- Лекции (слайды в github-репозитории)

Как устроен курс

github.com/lisyarus/graphics-course-slides/pdf

github.com/lisyarus/graphics-course-practice

- Лекции (слайды в github-репозитории)
- Практики:

Как устроен курс

github.com/lisyarus/graphics-course-slides/pdf

github.com/lisyarus/graphics-course-practice

- Лекции (слайды в github-репозитории)
- Практики:
 - Код-заготовка на C++ (в github-репозитории)

Как устроен курс

github.com/lisyarus/graphics-course-slides/pdf

github.com/lisyarus/graphics-course-practice

- Лекции (слайды в github-репозитории)
- Практики:
 - Код-заготовка на C++ (в github-репозитории)
 - Слайды с заданием (в github-репозитории)

Как устроен курс

github.com/lisyarus/graphics-course-slides/pdf

github.com/lisyarus/graphics-course-practice

- Лекции (слайды в github-репозитории)
- Практики:
 - Код-заготовка на C++ (в github-репозитории)
 - Слайды с заданием (в github-репозитории)
 - Сдача на занятии или отправкой кода

Как устроен курс

github.com/lisyarus/graphics-course-slides/pdf

github.com/lisyarus/graphics-course-practice

- Лекции (слайды в github-репозитории)
- Практики:
 - Код-заготовка на C++ (в github-репозитории)
 - Слайды с заданием (в github-репозитории)
 - Сдача на занятии или отправкой кода
- Домашние задания:

Как устроен курс

github.com/lisyarus/graphics-course-slides/pdf

github.com/lisyarus/graphics-course-practice

- Лекции (слайды в github-репозитории)
- Практики:
 - Код-заготовка на C++ (в github-репозитории)
 - Слайды с заданием (в github-репозитории)
 - Сдача на занятии или отправкой кода
- Домашние задания:
 - Слайды с заданием (в github-репозитории)

Как устроен курс

github.com/lisyarus/graphics-course-slides/pdf

github.com/lisyarus/graphics-course-practice

- Лекции (слайды в github-репозитории)
- Практики:
 - Код-заготовка на C++ (в github-репозитории)
 - Слайды с заданием (в github-репозитории)
 - Сдача на занятии или отправкой кода
- Домашние задания:
 - Слайды с заданием (в github-репозитории)
 - Сдача на практическом занятии

Как устроен курс

github.com/lisyarus/graphics-course-slides/pdf

github.com/lisyarus/graphics-course-practice

- Лекции (слайды в github-репозитории)
- Практики:
 - Код-заготовка на C++ (в github-репозитории)
 - Слайды с заданием (в github-репозитории)
 - Сдача на занятии или отправкой кода
- Домашние задания:
 - Слайды с заданием (в github-репозитории)
 - Сдача на практическом занятии
- Финальный проект
 - Большое, сложное, интересное задание по выбору

Как устроен курс

github.com/lisyarus/graphics-course-slides/pdf

github.com/lisyarus/graphics-course-practice

- Лекции (слайды в github-репозитории)
- Практики:
 - Код-заготовка на C++ (в github-репозитории)
 - Слайды с заданием (в github-репозитории)
 - Сдача на занятии или отправкой кода
- Домашние задания:
 - Слайды с заданием (в github-репозитории)
 - Сдача на практическом занятии
- Финальный проект
 - Большое, сложное, интересное задание по выбору
 - Сдача в конце курса

Баллы

Баллы

- ≈25 баллов – работа на практиках

Баллы

- **≈25 баллов** – работа на практиках
 - 1 балл – получилось хоть что-нибудь
 - 2 балла – получилось всё
 - 3 балла – получилось всё + доп. задание

Баллы

- **≈25 баллов** – работа на практиках
 - 1 балл – получилось хоть что-нибудь
 - 2 балл – получилось всё
 - 3 балла – получилось всё + доп. задание
 - Можно прислать код (или сдать лично) до конца следующей практики (через неделю)

Баллы

- ≈25 баллов – работа на практиках
 - 1 балл – получилось хоть что-нибудь
 - 2 балл – получилось всё
 - 3 балла – получилось всё + доп. задание
 - Можно прислать код (или сдать лично) до конца следующей практики (через неделю)
 - Можно прислать позже со штрафом в **1 балл**

Баллы

- **≈25 баллов** – работа на практиках
 - 1 балл – получилось хоть что-нибудь
 - 2 балл – получилось всё
 - 3 балла – получилось всё + доп. задание
 - Можно прислать код (или сдать лично) до конца следующей практики (через неделю)
 - Можно прислать позже со штрафом в **1 балл**
- **15 баллов** – каждое из 3х домашних заданий

Баллы

- **≈25 баллов** – работа на практиках
 - 1 балл – получилось хоть что-нибудь
 - 2 балл – получилось всё
 - 3 балла – получилось всё + доп. задание
 - Можно прислать код (или сдать лично) до конца следующей практики (через неделю)
 - Можно прислать позже со штрафом в **1 балл**
- **15 баллов** – каждое из 3х домашних заданий
 - Можно получить неполный балл

Баллы

- **≈25 баллов** – работа на практиках
 - 1 балл – получилось хоть что-нибудь
 - 2 балл – получилось всё
 - 3 балла – получилось всё + доп. задание
 - Можно прислать код (или сдать лично) до конца следующей практики (через неделю)
 - Можно прислать позже со штрафом в **1 балл**
- **15 баллов** – каждое из 3х домашних заданий
 - Можно получить неполный балл
 - Можно сдать после дня сдачи со штрафом в **50% баллов**

Баллы

- **≈25 баллов** – работа на практиках
 - 1 балл – получилось хоть что-нибудь
 - 2 балл – получилось всё
 - 3 балла – получилось всё + доп. задание
 - Можно прислать код (или сдать лично) до конца следующей практики (через неделю)
 - Можно прислать позже со штрафом в **1 балл**
- **15 баллов** – каждое из 3х домашних заданий
 - Можно получить неполный балл
 - Можно сдать после дня сдачи со штрафом в **50% баллов**
- **30+ баллов** – финальный проект

Оценка за курс

Оценка за курс

- Зачет: 50 и более баллов

Оценка за курс

- Зачет: 50 и более баллов
- Экзамен:
 - 50-59 баллов: E
 - 60-69 баллов: D
 - 70-79 баллов: C
 - 80-89 баллов: B
 - 90-100 баллов: A

Пререквизиты

Пререквизиты

- Программирование

Пререквизиты

- Программирование
 - Основы C++

Пререквизиты

- Программирование
 - Основы C++
 - Компилировать и запускать программы в удобной вам среде

Пререквизиты

- Программирование
 - Основы C++
 - Компилировать и запускать программы в удобной вам среде
- Математика

Пререквизиты

- Программирование
 - Основы C++
 - Компилировать и запускать программы в удобной вам среде
- Математика
 - Линейная алгебра (векторы, матрицы, умножение матриц, линейные системы, ортогональность)

Пререквизиты

- Программирование
 - Основы C++
 - Компилировать и запускать программы в удобной вам среде
- Математика
 - Линейная алгебра (векторы, матрицы, умножение матриц, линейные системы, ортогональность)
 - Аналитическая геометрия (координаты, уравнения кривых и поверхностей)

Что такое компьютерная графика?

Что такое компьютерная графика?

- Кинематограф, мультипликация

The Matrix Revolutions (2003)



Avatar (2009)



The Avengers (2012)



Klaus (2019)



Что такое компьютерная графика?

- Кинематограф, мультипликация
- Компьютерные игры

Space Invaders (1978)



Doom (1993)



Grand Theft Auto: Vice City (2002)



Civilization V (2010)



The Witcher 3: Wild Hunt (2015)



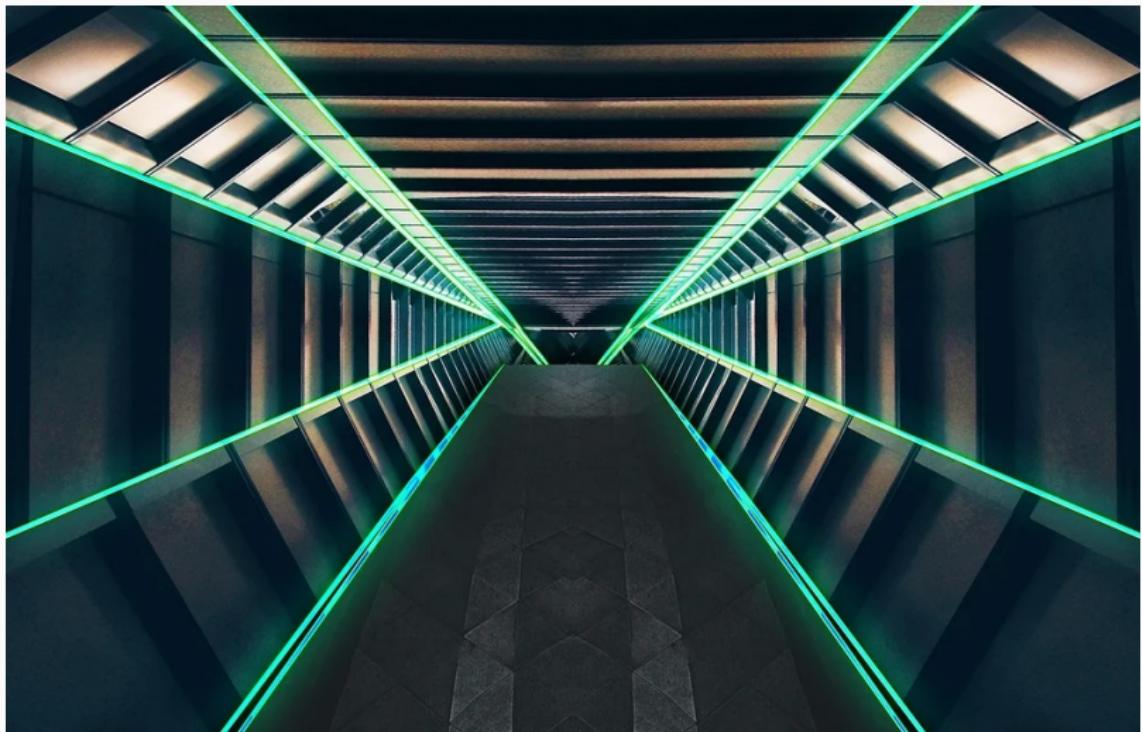
Cyberpunk 2077 (2020)

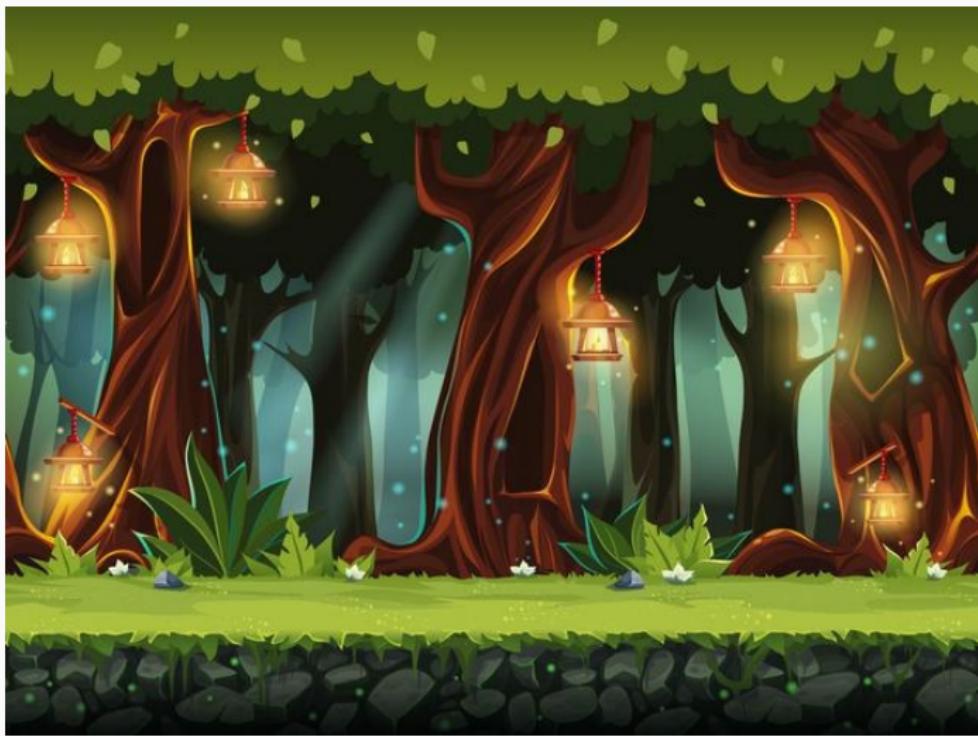


Что такое компьютерная графика?

- Кинематограф, мультипликация
- Компьютерные игры
- Рисунки, concept art







Что такое компьютерная графика?

- Кинематограф, мультипликация
- Компьютерные игры
- Рисунки, concept art
- Графический интерфейс

Mac OS Catalina



Windows 10

A

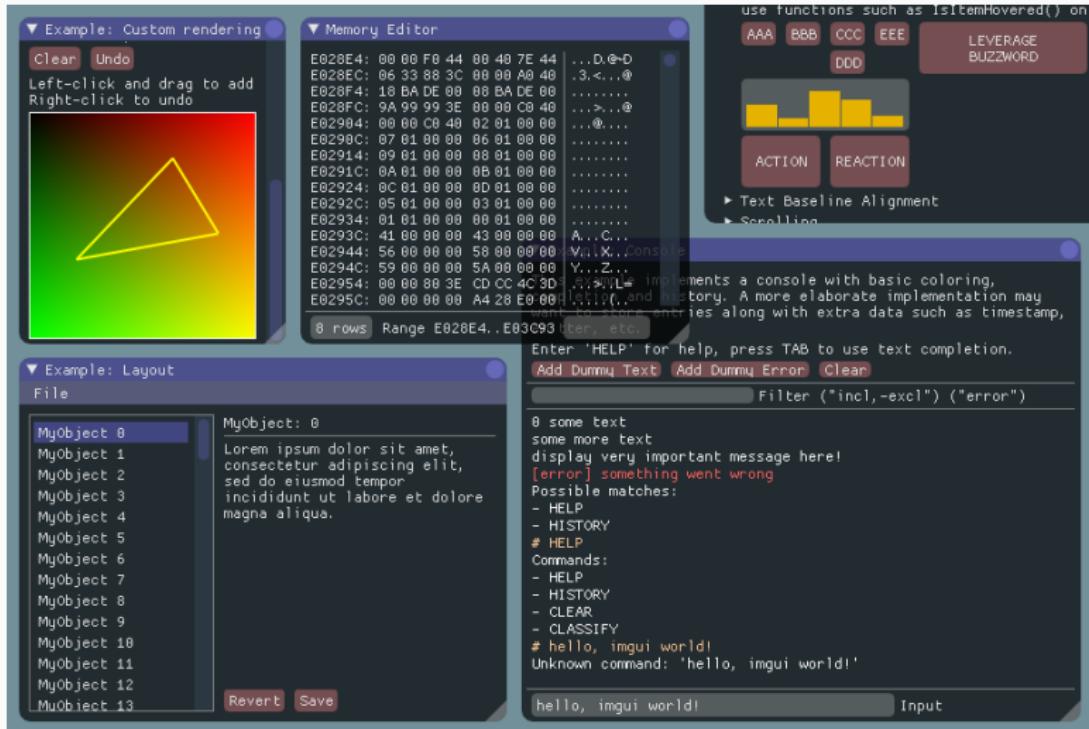
System accent color: #0078D4

Buttons	Calendar Date Picker	Combo box	Textbox
Enabled button	Label title mm/dd/yyyy	Label title Placeholder text	Label title Placeholder text
Disabled button	Hover mm/dd/yyyy	Hover Placeholder text	Hover Placeholder text
Toggle button	Disabled mm/dd/yyyy	Disabled Placeholder text	Disabled Placeholder text
Checkbox	February 2018	Microsoft	Typing
<input type="checkbox"/> Unchecked	Sun Mon Tue Wed Thu Fri Sat 31 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 1 2 3 4 5 6 7 8 9 10 11 12 13	Microsoft Windows Office	This is text.]
<input checked="" type="checkbox"/> Checked			
<input type="checkbox"/> Third state			
<input checked="" type="checkbox"/> Disabled			
Radio button	Microsoft	Windows	Password
<input type="radio"/> Unchecked	Microsoft	Windows	*****
<input checked="" type="radio"/> Checked	Windows	Office	
<input type="radio"/> Disabled			
Toggle switch	Off	Disabled Off	On
	<input type="radio"/>	Off	<input checked="" type="radio"/>
		Disabled Off	
	<input checked="" type="radio"/>	On	<input type="radio"/>
		Disabled On	

Europa Universalis 4



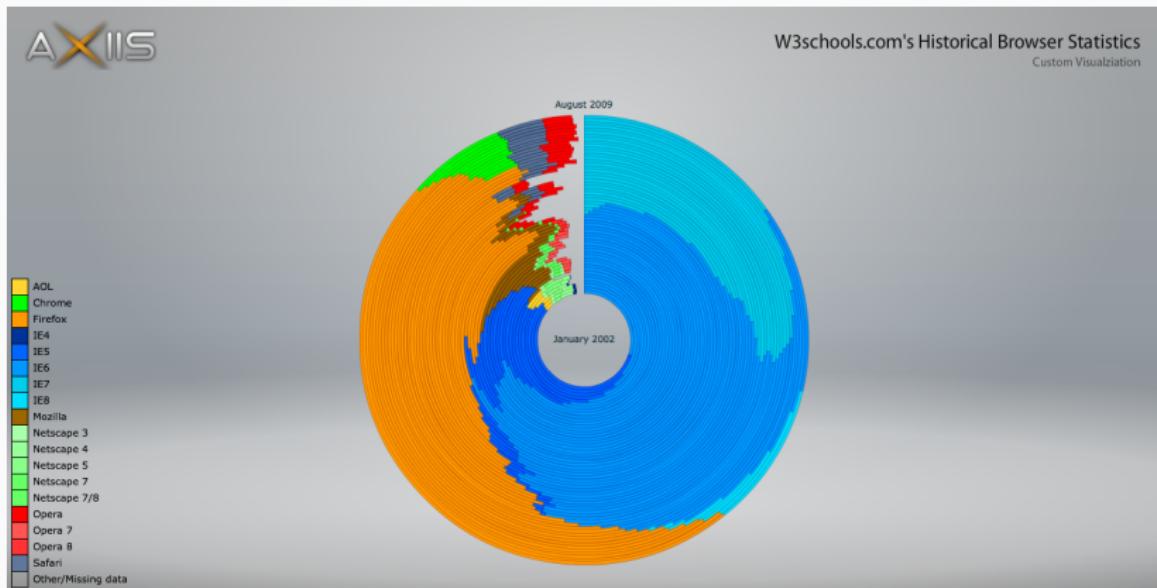
Dear ImGui



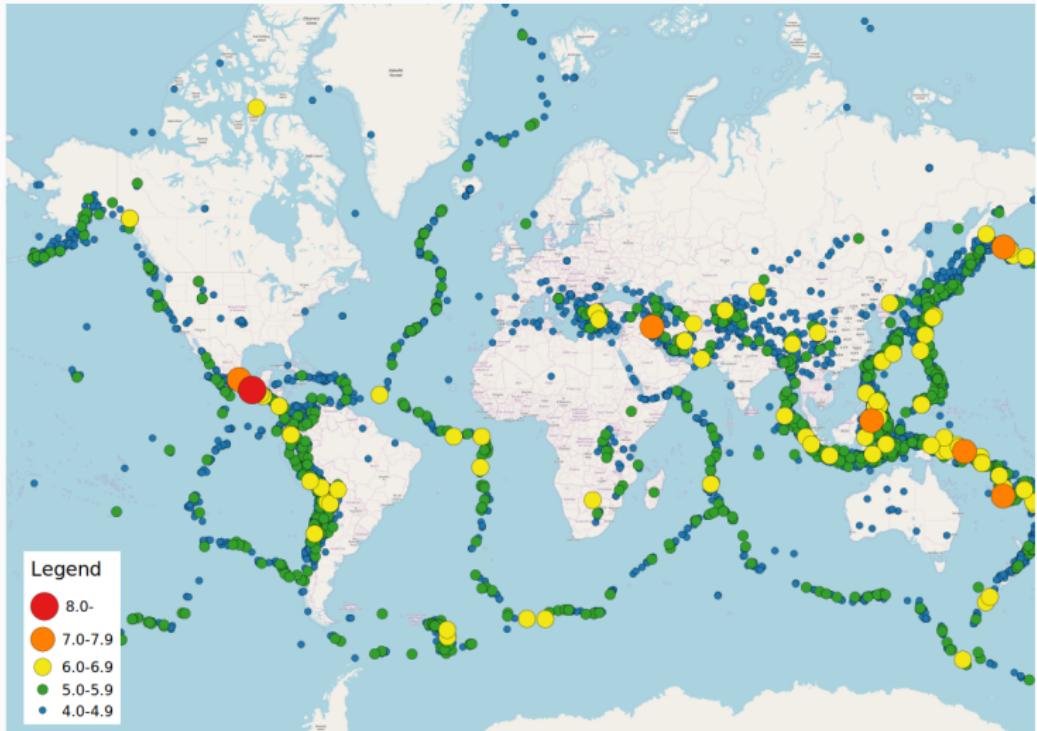
Что такое компьютерная графика?

- Кинематограф, мультипликация
- Компьютерные игры
- Рисунки, concept art
- Графический интерфейс
- Визуализация данных

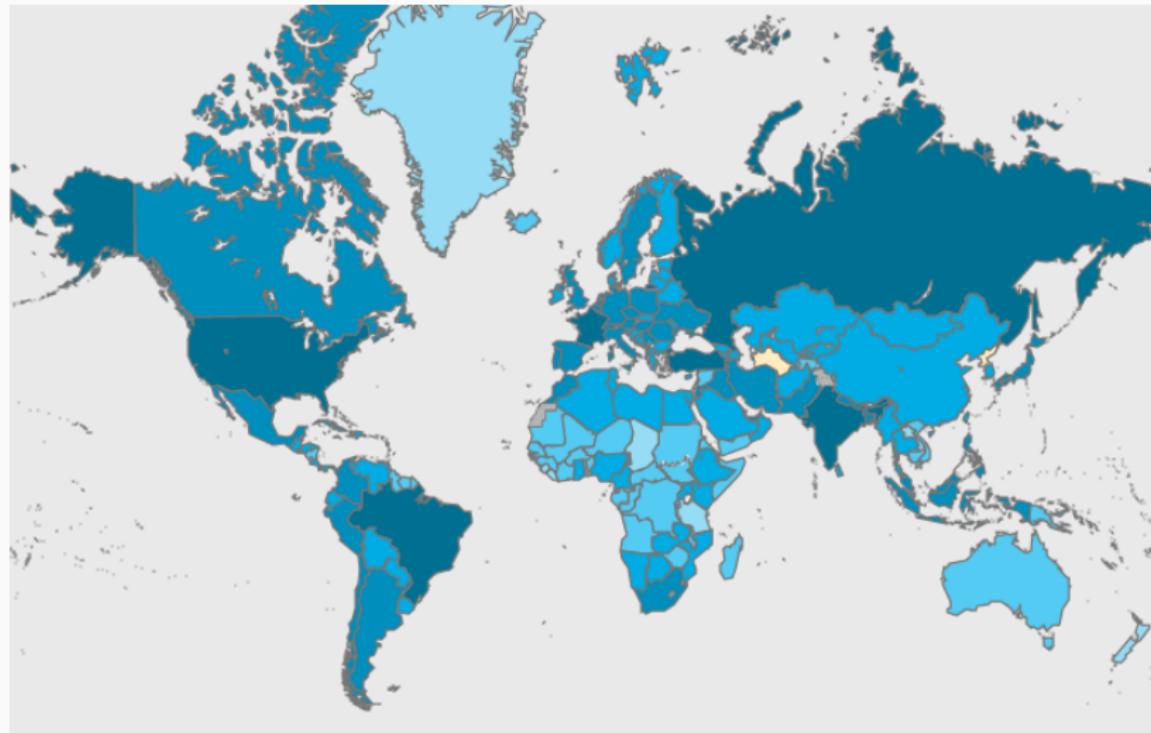
Популярность браузеров в 2002-2009



Карта землетрясений



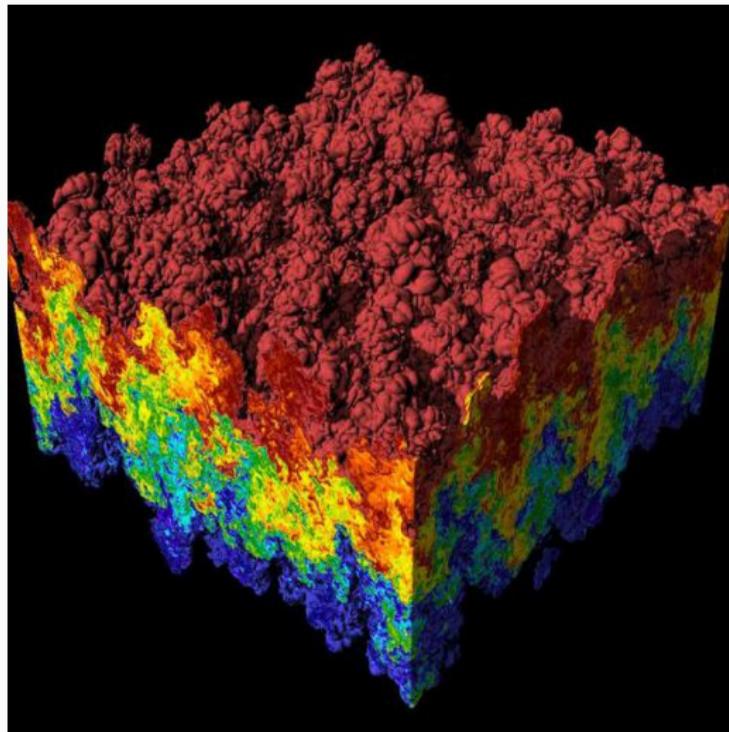
Количество случаев заражения COVID-19



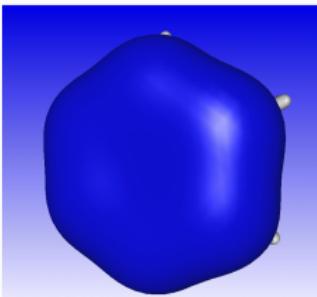
Что такое компьютерная графика?

- Кинематограф, мультипликация
- Компьютерные игры
- Рисунки, concept art
- Графический интерфейс
- Визуализация данных
- Научная визуализация

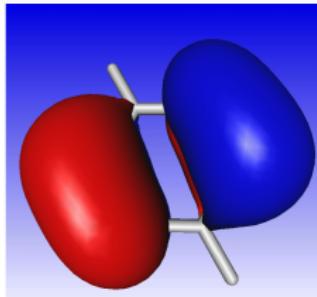
Неустойчивость Рэлея – Тейлора



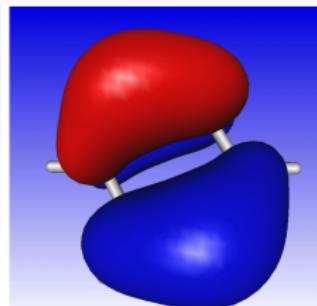
Молекулярные орбитали бензола



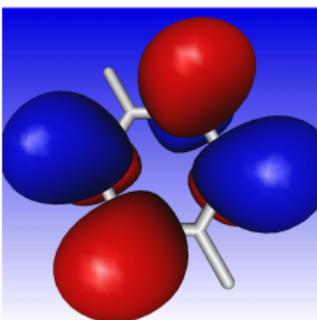
16



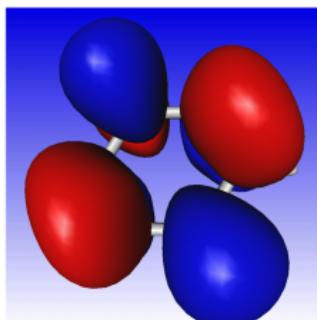
20



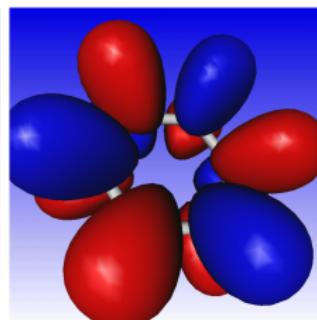
21



22

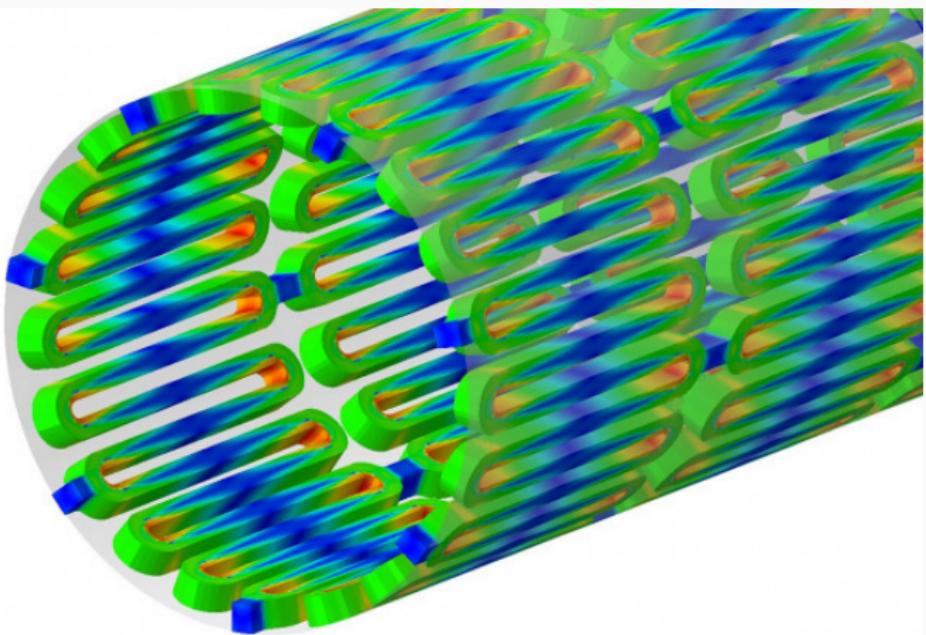


23



30

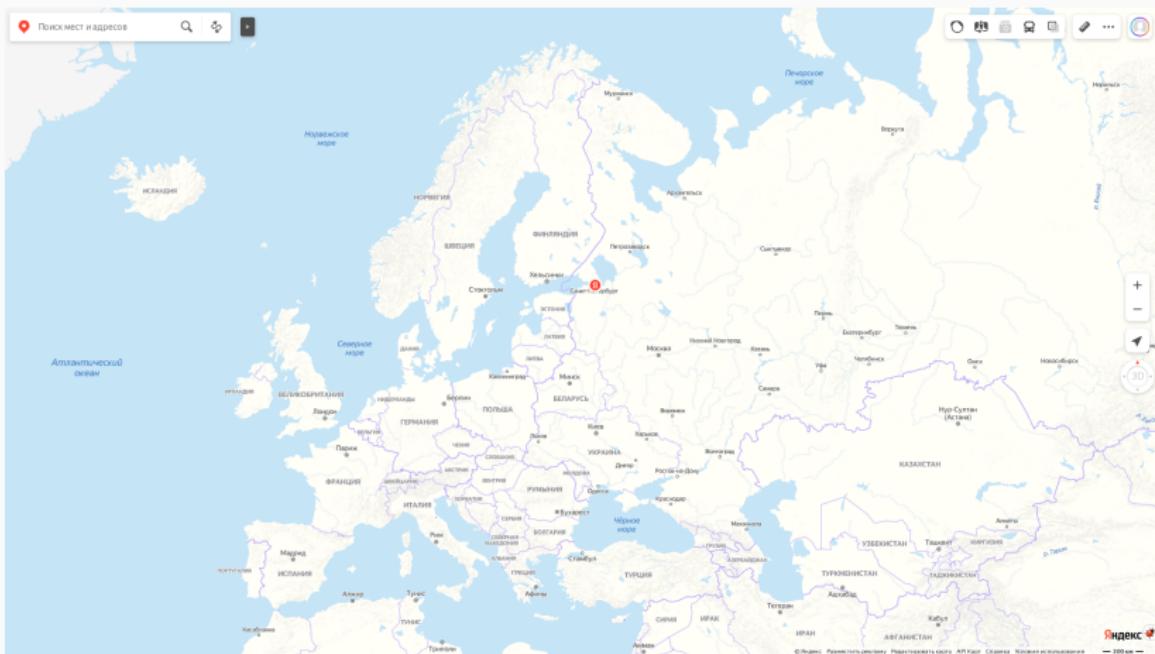
Симуляция напряжений в стене методом конечных элементов



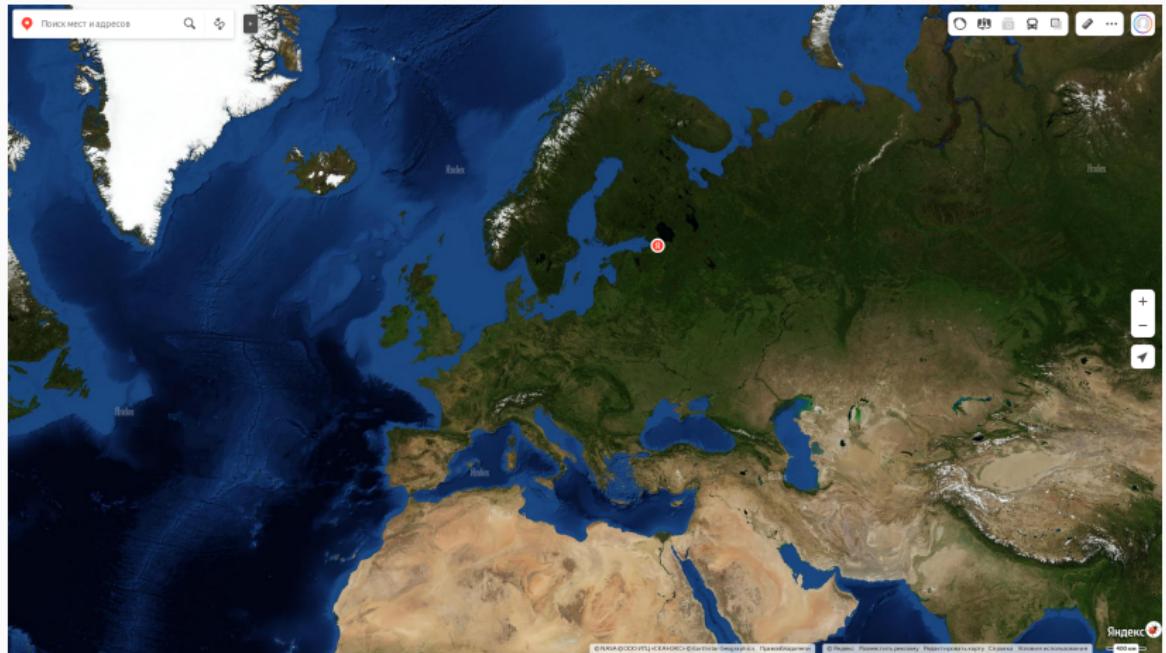
Что такое компьютерная графика?

- Кинематограф, мультипликация
- Компьютерные игры
- Рисунки, concept art
- Графический интерфейс
- Визуализация данных
- Научная визуализация
- Карты

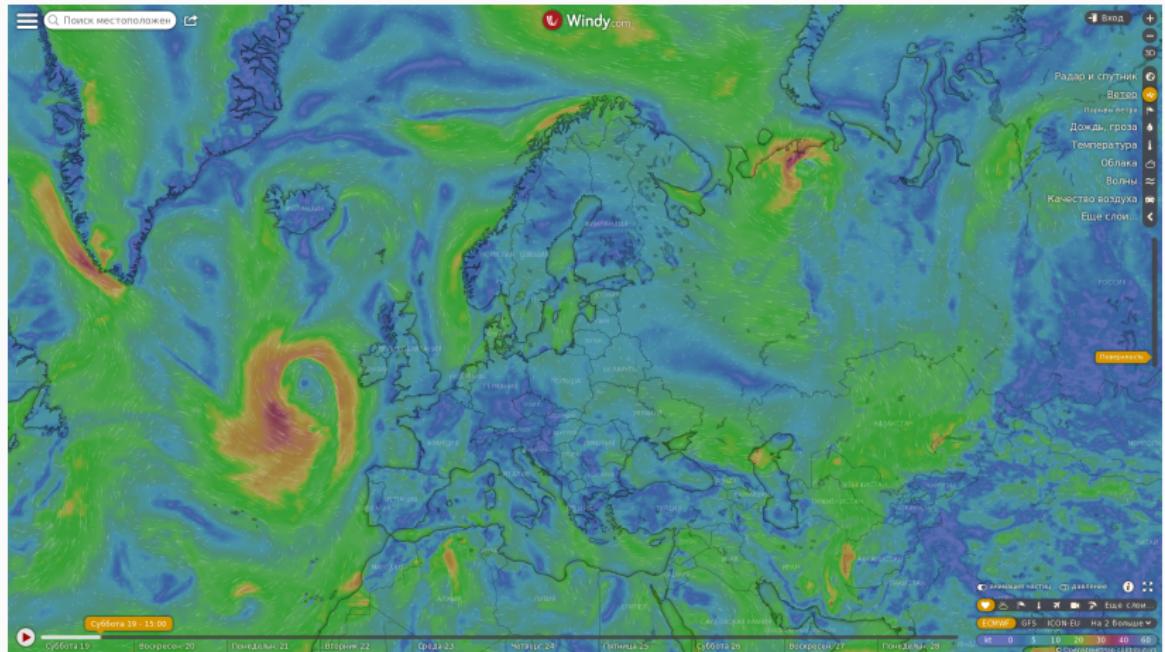
Схематическая карта



Спутниковая карта



Карта погоды



Что такое компьютерная графика?

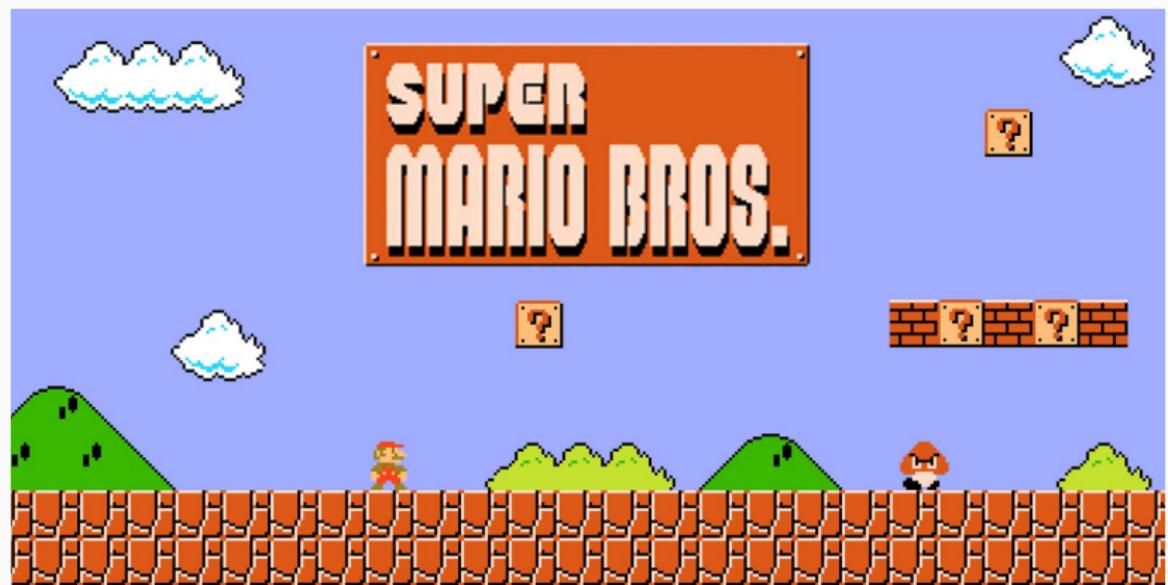
- Кинематограф, мультипликация
- Компьютерные игры
- Рисунки, concept art
- Графический интерфейс
- Визуализация данных
- Научная визуализация
- Карты
- И т.д.

Грубая и неточная классификация

Грубая и неточная классификация

- 2D / 3D

Super Mario Bros. (1983) - 2D



Red Dead Redemption 2 (2018) - 3D



Грубая и неточная классификация

- 2D / 3D

Грубая и неточная классификация

- 2D / 2.5D / 3D

Civilization III (2001) - 2.5D



Грубая и неточная классификация

- 2D / 2.5D / 3D

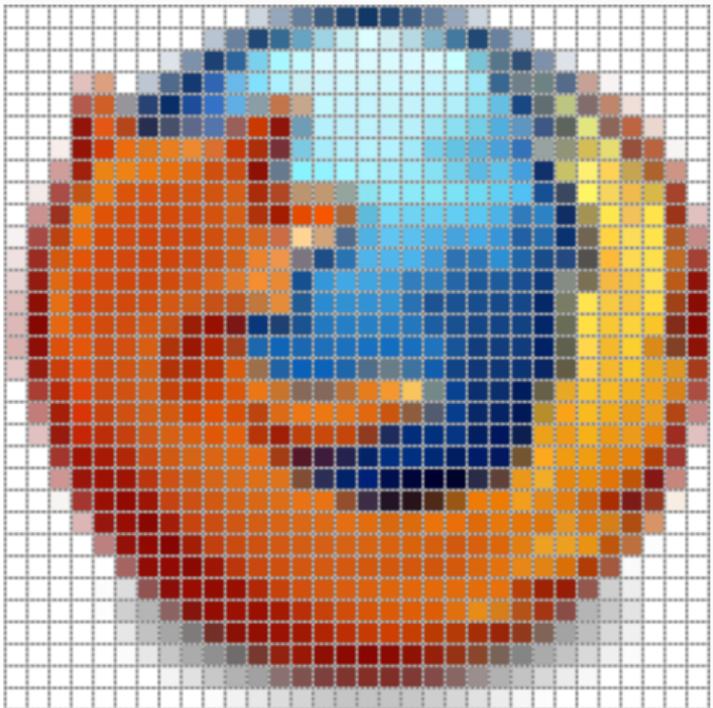
Грубая и неточная классификация

- 2D / 2.5D / 3D
- Векторная / растровая

Векторная графика



Растровая графика



Грубая и неточная классификация

- 2D / 2.5D / 3D
- Векторная / растровая

Грубая и неточная классификация

- 2D / 2.5D / 3D
- Векторная / растровая
- Realtime / offline

Грубая и неточная классификация

- 2D / 2.5D / 3D
- Векторная / растровая
- Realtime / near real-time / offline

Грубая и неточная классификация

- 2D / 2.5D / 3D
- Векторная / растровая
- Realtime / near real-time / offline
- Фотореалистичная / стилизованная

Грубая и неточная классификация

- 2D / 2.5D / 3D
- Векторная / растровая
- Realtime / near real-time / offline
- Фотореалистичная / стилизованная
- CPU / GPU

Чем мы будем заниматься?

- 2D / 2.5D / 3D
- Векторная / растровая
- Realtime / near real-time / offline
- Фотореалистичная / стилизованная
- CPU / GPU

Чем мы будем заниматься?

- 2D / 2.5D / 3D
- Векторная / растровая
- Realtime / near real-time / offline
- Фотореалистичная / стилизованная
- CPU / GPU

Чем мы будем заниматься?

- 2D / 2.5D / 3D
- Векторная / растровая
- Realtime / near real-time / offline
- Фотореалистичная / стилизованная
- CPU / GPU

Чем мы будем заниматься?

- 2D / 2.5D / 3D
- Векторная / растровая
- Realtime / near real-time / offline
- Фотореалистичная / стилизованная
- CPU / GPU

Чем мы будем заниматься?

- 2D / 2.5D / 3D
- Векторная / растровая
- Realtime / near real-time / offline
- Фотореалистичная / стилизованная
- CPU / GPU

Чем мы будем заниматься?

- 2D / 2.5D / 3D
- Векторная / растровая
- Realtime / near real-time / offline
- Фотореалистичная / стилизованная
- CPU / GPU

Чем мы будем заниматься?

Чем мы будем заниматься?

- Как пользоваться графическими API

Чем мы будем заниматься?

- Как пользоваться графическими API
- Как реализовывать графические движки

Чем мы будем заниматься?

- Как пользоваться графическими API
- Как реализовывать графические движки
- Как реализовывать графические эффекты

Чем мы будем заниматься?

- Как пользоваться графическими API
- Как реализовывать графические движки
- Как реализовывать графические эффекты
- Как их оптимизировать

Чем мы будем заниматься?

- Как пользоваться графическими API
- Как реализовывать графические движки
- Как реализовывать графические эффекты
- Как их оптимизировать
- Профессия *graphics engineer*

Где это пригодится?

Где это пригодится?

- Разработка игр и движков

Где это пригодится?

- Разработка игр и движков
- Разработка инструментов для художников/дизайнеров/архитекторов/etc

Где это пригодится?

- Разработка игр и движков
- Разработка инструментов для художников/дизайнеров/архитекторов/etc
- Разработка инструментов для научной визуализации/визуализации данных/etc

Где это пригодится?

- Разработка игр и движков
- Разработка инструментов для художников/дизайнеров/архитекторов/etc
- Разработка инструментов для научной визуализации/визуализации данных/etc
- Разработка картографических приложений

Где это пригодится?

- Разработка игр и движков
- Разработка инструментов для художников/дизайнеров/архитекторов/etc
- Разработка инструментов для научной визуализации/визуализации данных/etc
- Разработка картографических приложений
- Разработка движков графического интерфейса

Где это пригодится?

- Разработка игр и движков
- Разработка инструментов для художников/дизайнеров/архитекторов/etc
- Разработка инструментов для научной визуализации/визуализации данных/etc
- Разработка картографических приложений
- Разработка движков графического интерфейса
- И т.д.

Чем мы не будем заниматься?

Чем мы не будем заниматься?

- Учиться рисовать / моделировать и анимировать объекты / etc.

Чем мы не будем заниматься?

- Учиться рисовать / моделировать и анимировать объекты / etc.
 - Красивая картинка – движок + данные (текстуры, модели, частицы, etc, – assets)
 - Курс про **движок**

Чем мы не будем заниматься?

- Учиться рисовать / моделировать и анимировать объекты / etc.
 - Красивая картинка – движок + данные (текстуры, модели, частицы, etc, – assets)
 - Курс про *движок*
- Делать игры
 - Игра – гейм-дизайн + контент + графика + физика + механики + UI + аудио + сетевые компоненты + ...
 - Курс про *графику*

Примерный план курса

Примерный план курса

- Основы OpenGL
 - Как хранить данные на GPU
 - Как рисовать эти данные
 - Вершинные буферы, шейдеры, текстуры, фреймбуферы
 - Работа с камерой, перспективная проекция

Примерный план курса

- Основы OpenGL
 - Как хранить данные на GPU
 - Как рисовать эти данные
 - Вершинные буферы, шейдеры, текстуры, фреймбуферы
 - Работа с камерой, перспективная проекция
- Освещение
 - Теория
 - Модели освещения и материалов
 - Тени, отражения, ambient occlusion
 - Обработка большого количества источников света

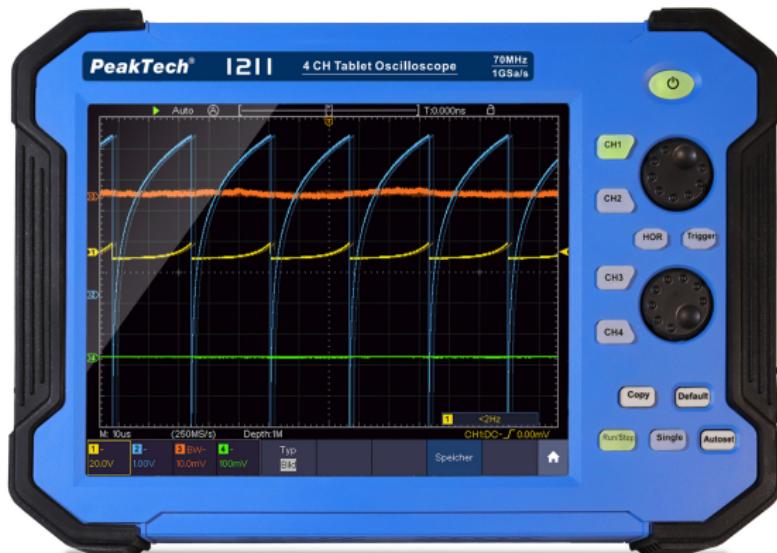
Примерный план курса

- Основы OpenGL
 - Как хранить данные на GPU
 - Как рисовать эти данные
 - Вершинные буферы, шейдеры, текстуры, фреймбуферы
 - Работа с камерой, перспективная проекция
- Освещение
 - Теория
 - Модели освещения и материалов
 - Тени, отражения, ambient occlusion
 - Обработка большого количества источников света
- Эффекты и оптимизации
 - Системы частиц (e.g. дым)
 - Скелетная анимация
 - Уровни детализации, frustum culling
 - Объёмный (volumetric) рендеринг
 - Рендеринг текста

Краткая история real-time компьютерной графики

1960-е: Осциллографы

Осциллограф



Краткая история real-time компьютерной графики

1960-е: Осциллографы

Краткая история real-time компьютерной графики

1960-е: Осциллографы

- Tennis For Two (1958)

Tennis For Two



Краткая история real-time компьютерной графики

1960-е: Осциллографы

- Tennis For Two (1958)

Краткая история real-time компьютерной графики

1960-е: Осциллографы

- Tennis For Two (1958)
- Spacewar! (1962, PDP-1)

PDP-1



Spacewar!



Краткая история real-time компьютерной графики

1960-е: Осциллографы

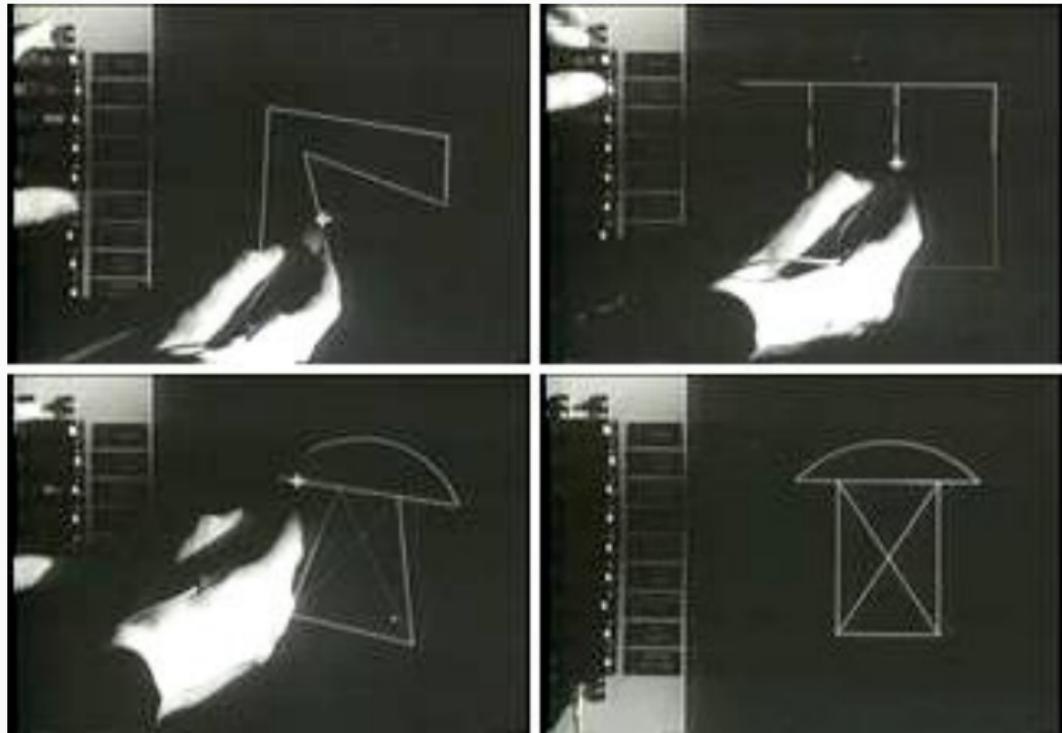
- Tennis For Two (1958)
- Spacewar! (1962, PDP-1)

Краткая история real-time компьютерной графики

1960-е: Осциллографы

- Tennis For Two (1958)
- Spacewar! (1962, PDP-1)
- Sketchpad (1963, TX-2, световое перо)

Sketchpad



archive.org/details/AlanKeyD1987

Краткая история real-time компьютерной графики

1960-е: Осциллографы

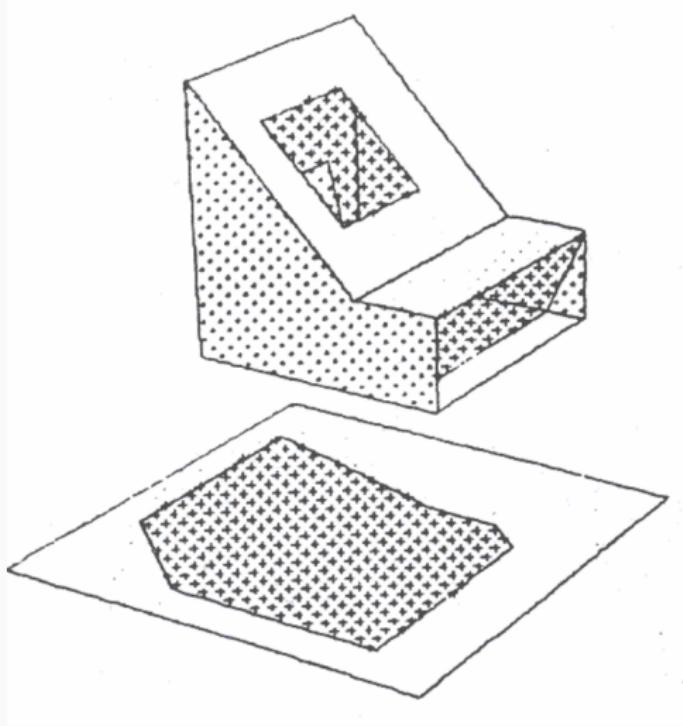
- Tennis For Two (1958)
- Spacewar! (1962, PDP-1)
- Sketchpad (1963, TX-2, световое перо)

Краткая история real-time компьютерной графики

1960-е: Осциллографы

- Tennis For Two (1958)
- Spacewar! (1962, PDP-1)
- Sketchpad (1963, TX-2, световое перо)
- Освещение и тени (Аппель, 1968)

Освещение (Аппель, 1968)



graphics.stanford.edu/courses/Appel.pdf

Краткая история real-time компьютерной графики

1960-е: Осциллографы

- Tennis For Two (1958)
- Spacewar! (1962, PDP-1)
- Sketchpad (1963, TX-2, световое перо)
- Освещение и тени (Аппель, 1968)

Краткая история real-time компьютерной графики

1960-е: Осциллографы

- Tennis For Two (1958)
- Spacewar! (1962, PDP-1)
- Sketchpad (1963, TX-2, световое перо)
- Освещение и тени (Аппель, 1968)
- Векторная графика довольно плохого качества: линии одного цвета
- Компьютеры довольно слабые (TX-2 занимал большую комнату)

Краткая история real-time компьютерной графики

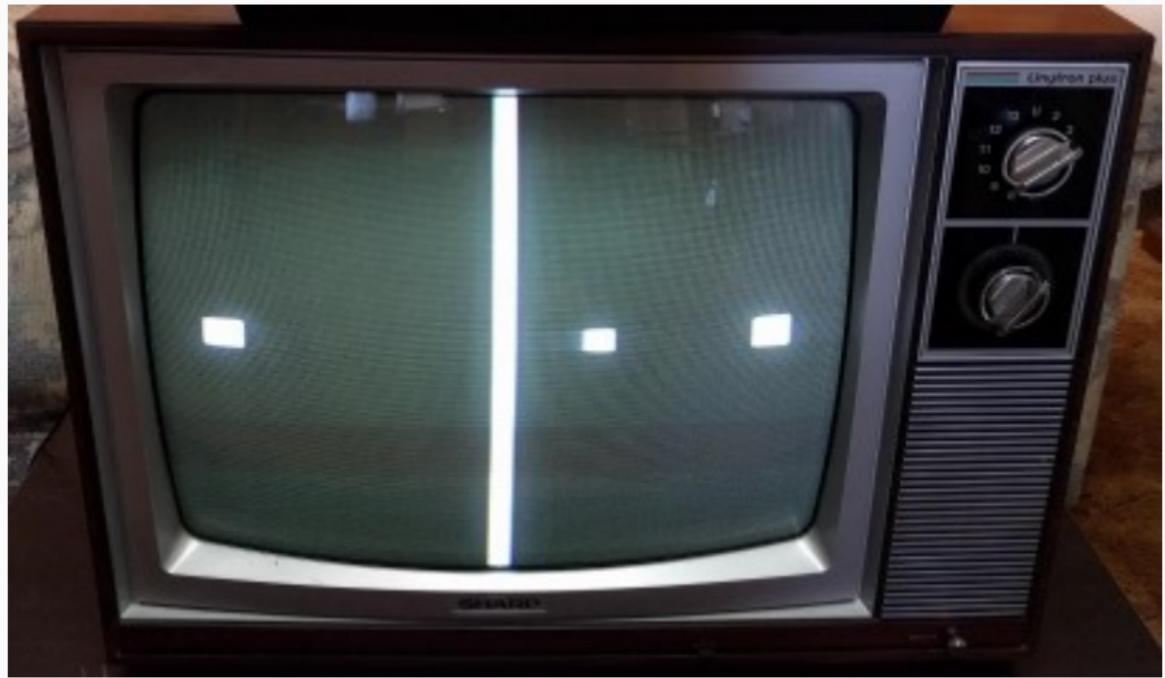
1970-е: Аркадные игры

Краткая история real-time компьютерной графики

1970-е: Аркадные игры

- Magnavox Odyssey (Magnavox, 1972) – первая игровая консоль, подключалась к телевизору (CRT)

Magnavox Odyssey



Краткая история real-time компьютерной графики

1970-е: Аркадные игры

- Magnavox Odyssey (Magnavox, 1972) – первая игровая консоль, подключалась к телевизору (CRT)

Краткая история real-time компьютерной графики

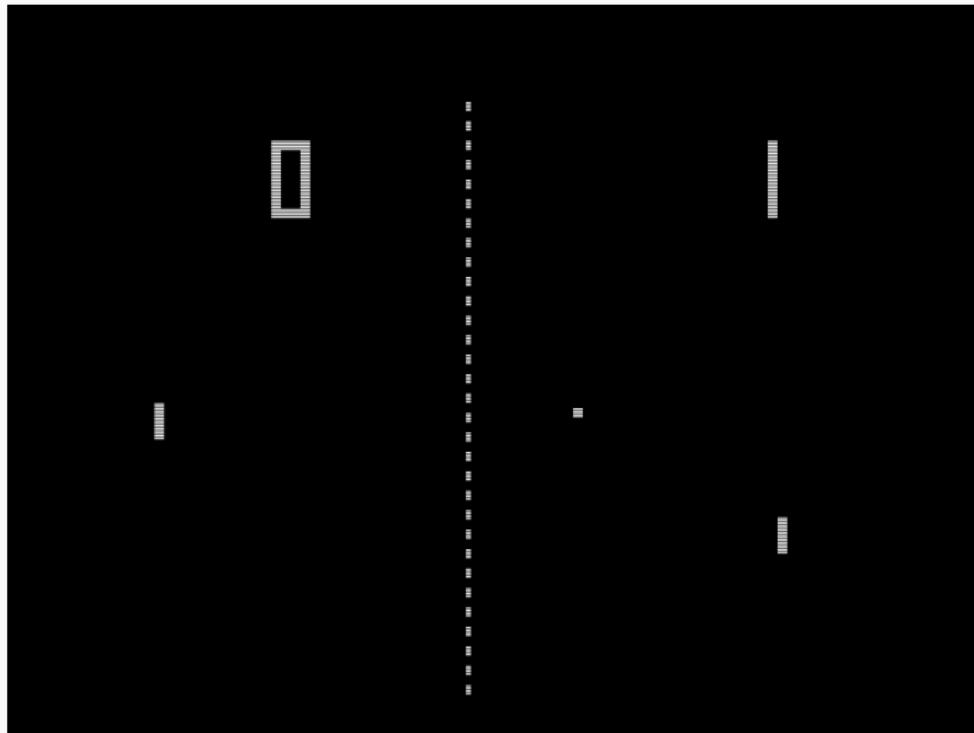
1970-е: Аркадные игры

- Magnavox Odyssey (Magnavox, 1972) – первая игровая консоль, подключалась к телевизору (CRT)
- Pong (Atari, 1972) – одна из первых аркадных игр

Pong



Pong



Краткая история real-time компьютерной графики

1970-е: Аркадные игры

- Magnavox Odyssey (Magnavox, 1972) – первая игровая консоль, подключалась к телевизору (CRT)
- Pong (Atari, 1972) – одна из первых аркадных игр

Краткая история real-time компьютерной графики

1970-е: Аркадные игры

- Magnavox Odyssey (Magnavox, 1972) – первая игровая консоль, подключалась к телевизору (CRT)
- Pong (Atari, 1972) – одна из первых аркадных игр
- Speed Race (Taito, 1974)

Speed Race



Краткая история real-time компьютерной графики

1970-е: Аркадные игры

- Magnavox Odyssey (Magnavox, 1972) – первая игровая консоль, подключалась к телевизору (CRT)
- Pong (Atari, 1972) – одна из первых аркадных игр
- Speed Race (Taito, 1974)

Краткая история real-time компьютерной графики

1970-е: Аркадные игры

- Magnavox Odyssey (Magnavox, 1972) – первая игровая консоль, подключалась к телевизору (CRT)
- Pong (Atari, 1972) – одна из первых аркадных игр
- Speed Race (Taito, 1974)
- Gun Fight (Taito, 1975)

Gun Fight



Краткая история real-time компьютерной графики

1970-е: Аркадные игры

- Magnavox Odyssey (Magnavox, 1972) – первая игровая консоль, подключалась к телевизору (CRT)
- Pong (Atari, 1972) – одна из первых аркадных игр
- Speed Race (Taito, 1974)
- Gun Fight (Taito, 1975)

Краткая история real-time компьютерной графики

1970-е: Аркадные игры

- Magnavox Odyssey (Magnavox, 1972) – первая игровая консоль, подключалась к телевизору (CRT)
- Pong (Atari, 1972) – одна из первых аркадных игр
- Speed Race (Taito, 1974)
- Gun Fight (Taito, 1975)
- Space Invaders (Taito, 1978)

Space Invaders



Краткая история real-time компьютерной графики

1970-е: Аркадные игры

- Magnavox Odyssey (Magnavox, 1972) – первая игровая консоль, подключалась к телевизору (CRT)
- Pong (Atari, 1972) – одна из первых аркадных игр
- Speed Race (Taito, 1974)
- Gun Fight (Taito, 1975)
- Space Invaders (Taito, 1978)

Краткая история real-time компьютерной графики

1970-е: Аркадные игры

- Magnavox Odyssey (Magnavox, 1972) – первая игровая консоль, подключалась к телевизору (CRT)
- Pong (Atari, 1972) – одна из первых аркадных игр
- Speed Race (Taito, 1974)
- Gun Fight (Taito, 1975)
- Space Invaders (Taito, 1978)
- Pac-Man (Namco, 1980)

Pac-Man



Space Invaders

Space Invaders

- Intel 8080, 8-bit, 2Mhz

Space Invaders

- Intel 8080, 8-bit, 2Mhz
- Экран 256x224, монохромный (1-bit)

Space Invaders

- Intel 8080, 8-bit, 2Mhz
- Экран 256x224, монохромный (1-bit)
- 8Kb ROM

Space Invaders

- Intel 8080, 8-bit, 2Mhz
- Экран 256x224, монохромный (1-bit)
- 8Kb ROM
- 8Kb RAM, из которых 7Kb занимал экран (framebuffer)

Краткая история real-time компьютерной графики

1970-е: Аркадные игры

- Magnavox Odyssey (Magnavox, 1972) – первая игровая консоль, подключалась к телевизору (CRT)
- Pong (Atari, 1972) – одна из первых аркадных игр
- Speed Race (Taito, 1974)
- Gun Fight (Taito, 1975)
- Space Invaders (Taito, 1978)
- Pac-Man (Namco, 1980)

Краткая история real-time компьютерной графики

1970-е: Аркадные игры

- Magnavox Odyssey (Magnavox, 1972) – первая игровая консоль, подключалась к телевизору (CRT)
- Pong (Atari, 1972) – одна из первых аркадных игр
- Speed Race (Taito, 1974)
- Gun Fight (Taito, 1975)
- Space Invaders (Taito, 1978)
- Pac-Man (Namco, 1980)
- Переход к растровой графике

Краткая история real-time компьютерной графики

1970-е: Аркадные игры

- Magnavox Odyssey (Magnavox, 1972) – первая игровая консоль, подключалась к телевизору (CRT)
- Pong (Atari, 1972) – одна из первых аркадных игр
- Speed Race (Taito, 1974)
- Gun Fight (Taito, 1975)
- Space Invaders (Taito, 1978)
- Pac-Man (Namco, 1980)
- Переход к растровой графике
- Разрешение экрана ограничено объёмами памяти

Краткая история real-time компьютерной графики

1980-е: Векторные аркады

Краткая история real-time компьютерной графики

1980-е: Векторные аркады

- Asteroids (Atari, 1979)

Asteroids



Краткая история real-time компьютерной графики

1980-е: Векторные аркады

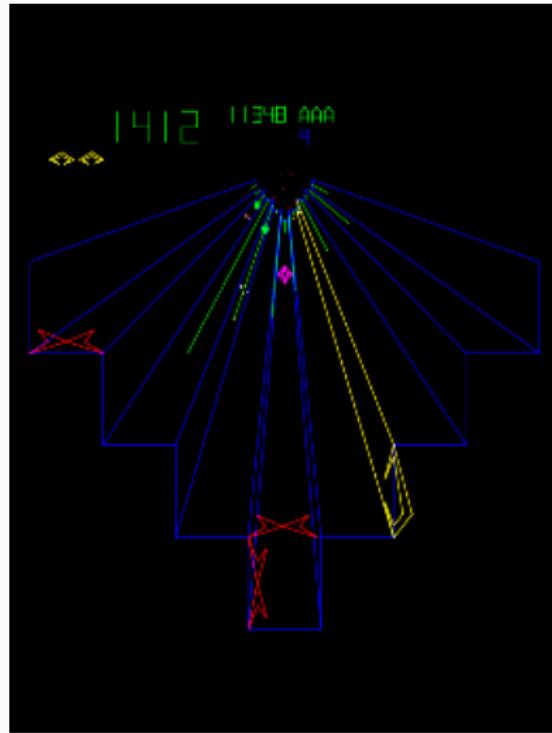
- Asteroids (Atari, 1979)

Краткая история real-time компьютерной графики

1980-е: Векторные аркады

- Asteroids (Atari, 1979)
- Tempest (Atari, 1981)

Tempest



Arcade Machines look WEIRD in Slow Mo – The Slow Mo Guys

Краткая история real-time компьютерной графики

1980-е: Векторные аркады

- Asteroids (Atari, 1979)
- Tempest (Atari, 1981)

Краткая история real-time компьютерной графики

1980-е: Векторные аркады

- Asteroids (Atari, 1979)
- Tempest (Atari, 1981)
- Star Wars (Atari, 1983)

Star Wars



Краткая история real-time компьютерной графики

1980-е: Векторные аркады

- Asteroids (Atari, 1979)
- Tempest (Atari, 1981)
- Star Wars (Atari, 1983)

Краткая история real-time компьютерной графики

1980-е: Векторные аркады

- Asteroids (Atari, 1979)
- Tempest (Atari, 1981)
- Star Wars (Atari, 1983)
- Требует специального оборудования (Atari's QuadraScan)

Краткая история real-time компьютерной графики

1980-е: Векторные аркады

- Asteroids (Atari, 1979)
- Tempest (Atari, 1981)
- Star Wars (Atari, 1983)
- Требует специального оборудования (Atari's QuadraScan)
- Может рисовать только линии

Краткая история real-time компьютерной графики

1980-е: 8-битные спрайтовые консоли

Краткая история real-time компьютерной графики

1980-е: 8-битные спрайтовые консоли

- Atari 2600 (Atari, 1977)

Atari 2600



Donkey Kong (Nintendo, 1981)



Pitfall! (Activision, 1982)



Краткая история real-time компьютерной графики

1980-е: 8-битные спрайтовые консоли

- Atari 2600 (Atari, 1977)

Краткая история real-time компьютерной графики

1980-е: 8-битные спрайтовые консоли

- Atari 2600 (Atari, 1977)
- NES (Nintendo, 1983)

Super Mario Bros. (Nintendo, 1985)



The Legend of Zelda. (Nintendo, 1986)



Краткая история real-time компьютерной графики

1980-е: 8-битные спрайтовые консоли

- Atari 2600 (Atari, 1977)
- NES (Nintendo, 1983)

Краткая история real-time компьютерной графики

1980-е: 8-битные спрайтовые консоли

- Atari 2600 (Atari, 1977)
- NES (Nintendo, 1983)
- Рисуют готовые изображения (спрайты) в указанных частях экрана

Краткая история real-time компьютерной графики

1980-е: 8-битные спрайтовые консоли

- Atari 2600 (Atari, 1977)
- NES (Nintendo, 1983)
- Рисуют готовые изображения (спрайты) в указанных частях экрана
- Не так требовательны к объёмам памяти

Краткая история real-time компьютерной графики

Конец 1980-х: 16-битные консоли и персональные
компьютеры

Краткая история real-time компьютерной графики

Конец 1980-х: 16-битные консоли и персональные компьютеры

- Sega Mega Drive (Sega, 1988)

Sonic the Hedgehog (Sega, 1991)



Краткая история real-time компьютерной графики

Конец 1980-х: 16-битные консоли и персональные компьютеры

- Sega Mega Drive (Sega, 1988)

Краткая история real-time компьютерной графики

Конец 1980-х: 16-битные консоли и персональные компьютеры

- Sega Mega Drive (Sega, 1988)
- Super NES (Nintendo, 1990)

Super Mario World (Nintendo, 1990)



Краткая история real-time компьютерной графики

Конец 1980-х: 16-битные консоли и персональные компьютеры

- Sega Mega Drive (Sega, 1988)
- Super NES (Nintendo, 1990)

Краткая история real-time компьютерной графики

Конец 1980-х: 16-битные консоли и персональные компьютеры

- Sega Mega Drive (Sega, 1988)
- Super NES (Nintendo, 1990)
- Больше памяти, быстрее процессоры

Краткая история real-time компьютерной графики

Конец 1980-х: 16-битные консоли и персональные компьютеры

- Sega Mega Drive (Sega, 1988)
- Super NES (Nintendo, 1990)
- Больше памяти, быстрее процессоры
- Поддерживают больше спрайтов и цветов

Краткая история real-time компьютерной графики

1990-е: Raycasting

Краткая история real-time компьютерной графики

1990-е: Raycasting

- Алгоритм рисования двумерных уровней в 3D

Краткая история real-time компьютерной графики

1990-е: Raycasting

- Алгоритм рисования двумерных уровней в 3D
- Wolfenstein 3D (id Software, 1992)

Wolfenstein 3D



Краткая история real-time компьютерной графики

1990-е: Raycasting

- Алгоритм рисования двумерных уровней в 3D
- Wolfenstein 3D (id Software, 1992)

Краткая история real-time компьютерной графики

1990-е: Raycasting

- Алгоритм рисования двумерных уровней в 3D
- Wolfenstein 3D (id Software, 1992)
- Doom (id Software, 1993)

Doom



Краткая история real-time компьютерной графики

1990-е: Raycasting

- Алгоритм рисования двумерных уровней в 3D
- Wolfenstein 3D (id Software, 1992)
- Doom (id Software, 1993)

Краткая история real-time компьютерной графики

1990-е: Raycasting

- Алгоритм рисования двумерных уровней в 3D
- Wolfenstein 3D (id Software, 1992)
- Doom (id Software, 1993)
- Quake (id Software, 1996)

Quake



Краткая история real-time компьютерной графики

1990-е: 32-битные консоли и компьютеры

Краткая история real-time компьютерной графики

1990-е: 32-битные консоли и компьютеры

- Графические процессоры с 3D графикой (Sony Playstation, Sega Saturn, Nintendo 64)

Краткая история real-time компьютерной графики

1990-е: 32-битные консоли и компьютеры

- Графические процессоры с 3D графикой (Sony Playstation, Sega Saturn, Nintendo 64)
- Умеют рисовать полигоны с текстурами и примитивным освещением (порядка нескольких тысяч за кадр)

Краткая история real-time компьютерной графики

1990-е: 32-битные консоли и компьютеры

- Графические процессоры с 3D графикой (Sony Playstation, Sega Saturn, Nintendo 64)
- Умеют рисовать полигоны с текстурами и примитивным освещением (порядка нескольких тысяч за кадр)
- Разрешения экрана до 640x480, 24-bit цвет

Краткая история real-time компьютерной графики

1990-е: 32-битные консоли и компьютеры

- Графические процессоры с 3D графикой (Sony Playstation, Sega Saturn, Nintendo 64)
- Умеют рисовать полигоны с текстурами и примитивным освещением (порядка нескольких тысяч за кадр)
- Разрешения экрана до 640x480, 24-bit цвет
- Virtua Racing (Sega, 1992)

Virtua Racing



Краткая история real-time компьютерной графики

1990-е: 32-битные консоли и компьютеры

- Графические процессоры с 3D графикой (Sony Playstation, Sega Saturn, Nintendo 64)
- Умеют рисовать полигоны с текстурами и примитивным освещением (порядка нескольких тысяч за кадр)
- Разрешения экрана до 640x480, 24-bit цвет
- Virtua Racing (Sega, 1992)

Краткая история real-time компьютерной графики

1990-е: 32-битные консоли и компьютеры

- Графические процессоры с 3D графикой (Sony Playstation, Sega Saturn, Nintendo 64)
- Умеют рисовать полигоны с текстурами и примитивным освещением (порядка нескольких тысяч за кадр)
- Разрешения экрана до 640x480, 24-bit цвет
- Virtua Racing (Sega, 1992)
- Tomb Raider (Core Design, 1996)

Tomb Raider



Краткая история real-time компьютерной графики

1990-е: 32-битные консоли и компьютеры

- Графические процессоры с 3D графикой (Sony Playstation, Sega Saturn, Nintendo 64)
- Умеют рисовать полигоны с текстурами и примитивным освещением (порядка нескольких тысяч за кадр)
- Разрешения экрана до 640x480, 24-bit цвет
- Virtua Racing (Sega, 1992)
- Tomb Raider (Core Design, 1996)

Краткая история real-time компьютерной графики

1990-е: 32-битные консоли и компьютеры

- Графические процессоры с 3D графикой (Sony Playstation, Sega Saturn, Nintendo 64)
- Умеют рисовать полигоны с текстурами и примитивным освещением (порядка нескольких тысяч за кадр)
- Разрешения экрана до 640x480, 24-bit цвет
- Virtua Racing (Sega, 1992)
- Tomb Raider (Core Design, 1996)
- Crash Bandicoot (Naughty Dog, 1996)

Crash Bandicoot



Краткая история real-time компьютерной графики

2000-е и 2010-е

Краткая история real-time компьютерной графики

2000-е и 2010-е

- Растут мощности как CPU, так и GPU

Краткая история real-time компьютерной графики

2000-е и 2010-е

- Растут мощности как CPU, так и GPU
- Растут доступные объёмы памяти

Краткая история real-time компьютерной графики

2000-е и 2010-е

- Растут мощности как CPU, так и GPU
- Растут доступные объёмы памяти
- Новые возможности GPU: шейдеры, рендеринг в текстуру, тесселяция

Краткая история real-time компьютерной графики

2000-е и 2010-е

- Растут мощности как CPU, так и GPU
- Растут доступные объёмы памяти
- Новые возможности GPU: шейдеры, рендеринг в текстуру, тесселяция
- Фотореалистичная графика

Ghost of Tsushima (Sucker Punch Productions, 2020)



Как использовать GPU?

GPU – Graphics Processing Unit

Как использовать GPU? Графические API

GPU – Graphics Processing Unit

Как использовать GPU? Графические API

GPU – Graphics Processing Unit

- Вендор-специфичные API (1980e – 1990e)

Как использовать GPU? Графические API

GPU – Graphics Processing Unit

- Вендор-специфичные API (1980е – 1990е)
- OpenGL (Silicon Graphics, 1992)

Как использовать GPU? Графические API

GPU – Graphics Processing Unit

- Вендор-специфичные API (1980е – 1990е)
- OpenGL (Silicon Graphics, 1992)
 - OpenGL 3.3 (Khronos Group, 2010)

Как использовать GPU? Графические API

GPU – Graphics Processing Unit

- Вендор-специфичные API (1980е – 1990е)
- OpenGL (Silicon Graphics, 1992)
 - OpenGL 3.3 (Khronos Group, 2010)
- DirectX (Microsoft, 1995)

Как использовать GPU? Графические API

GPU – Graphics Processing Unit

- Вендор-специфичные API (1980е – 1990е)
- OpenGL (Silicon Graphics, 1992)
 - OpenGL 3.3 (Khronos Group, 2010)
- DirectX (Microsoft, 1995)
 - DirectX 12 (Microsoft, 2015)

Как использовать GPU? Графические API

GPU – Graphics Processing Unit

- Вендор-специфичные API (1980е – 1990е)
- OpenGL (Silicon Graphics, 1992)
 - OpenGL 3.3 (Khronos Group, 2010)
- DirectX (Microsoft, 1995)
 - DirectX 12 (Microsoft, 2015)
- Metal (Apple, 2014)

Как использовать GPU? Графические API

GPU – Graphics Processing Unit

- Вендор-специфичные API (1980е – 1990е)
- OpenGL (Silicon Graphics, 1992)
 - OpenGL 3.3 (Khronos Group, 2010)
- DirectX (Microsoft, 1995)
 - DirectX 12 (Microsoft, 2015)
- Metal (Apple, 2014)
- Vulkan (Khronos Group, 2018)

Как использовать GPU? Графические API

GPU – Graphics Processing Unit

- Вендор-специфичные API (1980е – 1990е)
- OpenGL (Silicon Graphics, 1992)
 - OpenGL 3.3 (Khronos Group, 2010)
- DirectX (Microsoft, 1995)
 - DirectX 12 (Microsoft, 2015)
- Metal (Apple, 2014)
- Vulkan (Khronos Group, 2018)
- WebGPU (W3C, working draft)

Как использовать GPU? Графические API

GPU – Graphics Processing Unit

- Вендор-специфичные API (1980е – 1990е)
- OpenGL (Silicon Graphics, 1992)
 - OpenGL 3.3 (Khronos Group, 2010)
- DirectX (Microsoft, 1995)
 - DirectX 12 (Microsoft, 2015)
- Metal (Apple, 2014)
- Vulkan (Khronos Group, 2018)
- WebGPU (W3C, working draft)

Современные API

Как использовать GPU? Графические API

GPU – Graphics Processing Unit

- Вендор-специфичные API (1980е – 1990е)
- OpenGL (Silicon Graphics, 1992)
 - OpenGL 3.3 (Khronos Group, 2010)
- DirectX (Microsoft, 1995)
 - DirectX 12 (Microsoft, 2015)
- Metal (Apple, 2014)
- Vulkan (Khronos Group, 2018)
- WebGPU (W3C, working draft)

Современные API

Как использовать GPU? API общего назначения (GPGPU):

GPGPU – General-Purpose Graphics Processing Unit

Как использовать GPU? API общего назначения (GPGPU):

GPGPU – General-Purpose Graphics Processing Unit

- CUDA (Nvidia, 2007)

Как использовать GPU? API общего назначения (GPGPU):

GPGPU – General-Purpose Graphics Processing Unit

- CUDA (Nvidia, 2007)
- DirectX 11 DirectCompute (Microsoft, 2008)

Как использовать GPU? API общего назначения (GPGPU):

GPGPU – General-Purpose Graphics Processing Unit

- CUDA (Nvidia, 2007)
- DirectX 11 DirectCompute (Microsoft, 2008)
- OpenCL (Khronos Group, 2009)

Как использовать GPU? API общего назначения (GPGPU):

GPGPU – General-Purpose Graphics Processing Unit

- CUDA (Nvidia, 2007)
- DirectX 11 DirectCompute (Microsoft, 2008)
- OpenCL (Khronos Group, 2009)
- OpenGL 4.3 Compute Shaders (Khronos Group, 2012)

Как использовать GPU? API общего назначения (GPGPU):

GPGPU – General-Purpose Graphics Processing Unit

- CUDA (Nvidia, 2007)
- DirectX 11 DirectCompute (Microsoft, 2008)
- OpenCL (Khronos Group, 2009)
- OpenGL 4.3 Compute Shaders (Khronos Group, 2012)
- Metal Compute Shaders (Apple, 2014)

Как использовать GPU? API общего назначения (GPGPU):

GPGPU – General-Purpose Graphics Processing Unit

- CUDA (Nvidia, 2007)
- DirectX 11 DirectCompute (Microsoft, 2008)
- OpenCL (Khronos Group, 2009)
- OpenGL 4.3 Compute Shaders (Khronos Group, 2012)
- Metal Compute Shaders (Apple, 2014)
- Vulkan Compute Shaders (Khronos Group, 2018)

Как использовать GPU? API общего назначения (GPGPU):

GPGPU – General-Purpose Graphics Processing Unit

- CUDA (Nvidia, 2007)
- DirectX 11 DirectCompute (Microsoft, 2008)
- OpenCL (Khronos Group, 2009)
- OpenGL 4.3 Compute Shaders (Khronos Group, 2012)
- Metal Compute Shaders (Apple, 2014)
- Vulkan Compute Shaders (Khronos Group, 2018)
- WebGPU Compute Shaders (W3C, 2021)

Почему OpenGL 3.3?

Почему OpenGL 3.3?

- Широкая поддержка: интегрированные GPU, встраиваемые устройства, телефоны, web, некоторые игровые приставки

Почему OpenGL 3.3?

- Широкая поддержка: интегрированные GPU, встраиваемые устройства, телефоны, web, некоторые игровые приставки
- Поддерживает всё, что нам нужно

Почему OpenGL 3.3?

- Широкая поддержка: интегрированные GPU, встраиваемые устройства, телефоны, web, некоторые игровые приставки
- Поддерживает всё, что нам нужно
- +/- Кроссплатформенность

Почему OpenGL 3.3?

- Широкая поддержка: интегрированные GPU, встраиваемые устройства, телефоны, web, некоторые игровые приставки
- Поддерживает всё, что нам нужно
- +/- Кроссплатформенность (спасибо, Apple)

Почему OpenGL 3.3?

- Широкая поддержка: интегрированные GPU, встраиваемые устройства, телефоны, web, некоторые игровые приставки
- Поддерживает всё, что нам нужно
- +/- Кроссплатформенность (спасибо, Apple)
- Низкий порог входления (в сравнении с более современными API)

Почему OpenGL 3.3?

- Широкая поддержка: интегрированные GPU, встраиваемые устройства, телефоны, web, некоторые игровые приставки
- Поддерживает всё, что нам нужно
 - +/- Кроссплатформенность (спасибо, Apple)
 - Низкий порог входления (в сравнении с более современными API)
 - Достаточно старый
 - Много вспомогательных библиотек
 - Известны best practices
 - Известны все грабли

Почему OpenGL 3.3?

- Широкая поддержка: интегрированные GPU, встраиваемые устройства, телефоны, web, некоторые игровые приставки
- Поддерживает всё, что нам нужно
 - +/- Кроссплатформенность (спасибо, Apple)
 - Низкий порог входления (в сравнении с более современными API)
 - Достаточно старый
 - Много вспомогательных библиотек
 - Известны best practices
 - Известны все грабли (их много)

Почему OpenGL 3.3?

Почему OpenGL 3.3?

- Крупные движки переписывают на Vulkan / DirectX 12

Почему OpenGL 3.3?

- Крупные движки переписывают на Vulkan / DirectX 12
- Не у всех есть на это ресурсы

Почему OpenGL 3.3?

- Крупные движки переписывают на Vulkan / DirectX 12
- Не у всех есть на это ресурсы
- Не всем нужна самая крутая графика и производительность (работает в 60 fps – и ладно)

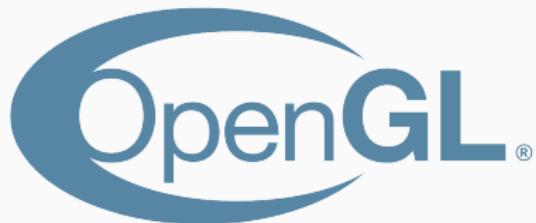
Почему OpenGL 3.3?

- Крупные движки переписывают на Vulkan / DirectX 12
- Не у всех есть на это ресурсы
- Не всем нужна самая крутая графика и производительность (работает в 60 fps – и ладно)
- OpenGL всё ещё широко используется

Почему OpenGL 3.3?

- Крупные движки переписывают на Vulkan / DirectX 12
- Не у всех есть на это ресурсы
- Не всем нужна самая крутая графика и производительность (работает в 60 fps – и ладно)
- OpenGL всё ещё широко используется
- Все основные концепции OpenGL (шейдеры, атрибуты вершин, буферы с данными, текстуры, ...) есть в любом графическом API ⇒ изучение OpenGL поможет в изучении более современных API

История графических API: OpenGL 1.0 (1992)



```
for object in scene.objects:  
    glBegin(GL_TRIANGLES)  
        for triangle in object.triangles:  
            for vertex in triangle.vertices:  
                glColor3f(vertex.color)  
                glNormal3f(vertex.normal)  
                glVertex3f(vertex.position)  
    glEnd(GL_TRIANGLES)
```

История графических API: OpenGL 1.0 (1992)

- Данные хранятся в памяти CPU

История графических API: OpenGL 1.0 (1992)

- Данные хранятся в памяти CPU
- Несколько OpenGL-вызовов на каждую вершину

История графических API: OpenGL 1.0 (1992)

- Данные хранятся в памяти CPU
- Несколько OpenGL-вызовов на каждую вершину
 - GPU становится быстрее \Rightarrow основное время тратится не на рисование, а на накладные расходы самих OpenGL-вызовов

История графических API: OpenGL 1.0 (1992)

- Как менять положение объектов и камеры?

История графических API: OpenGL 1.0 (1992)

- Как менять положение объектов и камеры?
- \Rightarrow Матрицы преобразований
`(glMatrixMode(), glLoadMatrix())`

История графических API: OpenGL 1.0 (1992)

- Как менять положение объектов и камеры?
- \Rightarrow Матрицы преобразований
(`glMatrixMode()`, `glLoadMatrix()`)
- Fixed-function pipeline: настраиваемая, но не расширяемая последовательность операций
(применить матрицы к входным данным, нарисовать треугольник на экране, выполнить тест глубины, ...)

История графических API: OpenGL 1.0 (1992)

- Как менять положение объектов и камеры?
- \Rightarrow Матрицы преобразований
(`glMatrixMode()`, `glLoadMatrix()`)
- Fixed-function pipeline: настраиваемая, но не расширяемая последовательность операций
(применить матрицы к входным данным, нарисовать треугольник на экране, выполнить тест глубины, ...)
- Асинхронный API: команды выполняются на GPU
когда-нибудь

История графических API: OpenGL 1.1 (1997)

- Vertex array – спецификация формата и расположения вершин
 - Сказать, где находятся вершины одной командой `glVertexPointer`
 - Нарисовать все вершины одной командой `glDrawArrays`
 - Вершины всё ещё хранятся на CPU

История графических API: OpenGL 1.1 (1997)

- Vertex array – спецификация формата и расположения вершин
 - Сказать, где находятся вершины одной командой `glVertexPointer`
 - Нарисовать все вершины одной командой `glDrawArrays`
 - Вершины всё ещё хранятся на CPU
- Текстуры – изображения в памяти GPU, натягиваемые на полигоны

История графических API: OpenGL 1.1 (1997)

```
// на старте
for object in scene.objects:
    object.createVertexArray(object.vertices)
    object.createTexture()

// при рендеринге
for object in scene.objects:
    glBindTexture(GL_TEXTURE_2D, object.texture)
    glBindVertexArray(object.vertexArray)
    glDrawArrays(object.vertexCount)
```

История графических API: OpenGL 1.2 - 1.4 (1998 - 2002)

- В текстурах можно записать очень много интересного, помимо цвета: normal map, material map, bump map
- Хочется выполнять сложные вычисления на каждый пиксель

История графических API: OpenGL 1.2 - 1.4 (1998 - 2002)

- В текстурах можно записать очень много интересного, помимо цвета: normal map, material map, bump map
- Хочется выполнять сложные вычисления на каждый пиксель
- \Rightarrow Texture environments – зачатки программируемости GPU (шейдеров)

История графических API: OpenGL 1.5 (2003)

- Vertex buffer – возможность хранить вершины в памяти GPU

История графических API: OpenGL 1.5 (2003)

- Vertex buffer – возможность хранить вершины в памяти GPU

```
// на старте
for object in scene.objects:
    object.createVertexBuffer(object.vertices)
    object.createVertexArray(object.vertexBuffer)
    object.createTexture()

// при рендеринге
for object in scene.objects:
    glBindTexture(GL_TEXTURE_2D, object.texture)
    glBindVertexArray(object.vertexArray)
    glDrawArrays(object.vertexCount)
```

История графических API: OpenGL 2.0 (2004)

- Шейдеры – программы на С-подобном языке GLSL, компилируемые под конкретную GPU
- Заменяют fixed-function pipeline
 - Необходимые части fixed-function pipeline остаются (растеризация, тест глубины, etc)

История графических API: OpenGL 2.0 (2004)

```
// на старте
for object in scene.objects:
    object.createVertexBuffer(object.vertices)
    object.createVertexArray(object.vertexBuffer)
    object.createTexture()
    object.createShaderProgram()

// при рендеринге
for object in scene.objects:
    glBindTexture(GL_TEXTURE_2D, object.texture)
    glBindVertexArray(object.vertexArray)
    glUseProgram(object.shaderProgram)
    glDrawArrays(object.vertexCount)
```

История графических API: OpenGL 2.0 (2004)

```
// на старте
for object in scene.objects:
    object.createVertexBuffer(object.vertices)
    object.createVertexArray(object.vertexBuffer)
    object.createTexture()
    object.createShaderProgram()

// при рендеринге
for object in scene.objects:
    glBindTexture(GL_TEXTURE_2D, object.texture)
    glBindVertexArray(object.vertexArray)
    glUseProgram(object.shaderProgram)
    glDrawArrays(object.vertexCount)
```

- Примерно так будет выглядеть наш код

История графических API: OpenGL 3.0 (2008)

- Огромная часть API объявлена deprecated

История графических API: OpenGL 3.0 (2008)

- Огромная часть API объявлена deprecated
 - Immediate-mode рисование – `glBegin/glEnd`
 - Хранение данных на CPU – `glVertexPointer`, ...
 - Матрицы преобразований – `glLoadMatrix`, ...

История графических API: OpenGL 3.0 (2008)

- Огромная часть API объявлена deprecated
 - Immediate-mode рисование – `glBegin/glEnd`
 - Хранение данных на CPU – `glVertexPointer`, ...
 - Матрицы преобразований – `glLoadMatrix`, ...
- Transform feedback – возможность записать результат работы шейдеров обратно в вершинный буфер
 - Зачатки GPGPU

История графических API: OpenGL 3.1 (2009)

- Объявленные deprecated возможности удалены

История графических API: OpenGL 3.1 (2009)

- Объявленные *deprecated* возможности удалены
- *Instanced rendering* – нарисовать много копий одного объекта в разных местах одной командой

История графических API: OpenGL 3.2 (2009)

- Механизм профилей
 - Core profile
 - Обязан поддерживаться
 - Только функционал конкретной версии OpenGL
 - Compatibility profile
 - Не обязан поддерживаться
 - Функционал этой и всех предыдущих версий OpenGL

История графических API: OpenGL 3.2 (2009)

- Механизм профилей
 - Core profile
 - Обязан поддерживаться
 - Только функционал конкретной версии OpenGL
 - Compatibility profile
 - Не обязан поддерживаться
 - Функционал этой и всех предыдущих версий OpenGL
 - Мы будем использовать core profile

История графических API: OpenGL 3.2 (2009)

- Механизм профилей
 - Core profile
 - Обязан поддерживаться
 - Только функционал конкретной версии OpenGL
 - Compatibility profile
 - Не обязан поддерживаться
 - Функционал этой и всех предыдущих версий OpenGL
 - Мы будем использовать core profile
- Геометрические шейдеры – возможность менять тип геометрии и количество вершин на лету (используются для систем частиц, травы, etc)

История графических API: OpenGL 4.0 (2010)

- Шейдеры тесселяции – увеличивают детализацию геометрии на лету
 - Гораздо меньше возможностей, чем у геометрических шейдеров, зато быстрее

История графических API: OpenGL 4.0 (2010)

- Шейдеры тесселяции – увеличивают детализацию геометрии на лету
 - Гораздо меньше возможностей, чем у геометрических шейдеров, зато быстрее
- Indirect drawing – можно вычислять количество вершин и их расположение в памяти на лету на GPU, и использовать вычисленные значения для команд рисования

История графических API: OpenGL 4.1 - 4.7 (2010 - 2017)

- Compute shaders – настоящее GPGPU внутри OpenGL
- Проработка и детализация API
- Атомарные операции в шейдерах
- Вливание расширений в стандарт OpenGL
- khronos.org/opengl/wiki/History_of_OpenGL



- 700 строк кода, чтобы нарисовать один треугольник



- 700 строк кода, чтобы нарисовать один треугольник
- Крайне низкоуровневый API
- Последовательности команд для выполнения на GPU (command queues) в явном виде
 - Можно распараллелить генерацию command queues на несколько CPU



- 700 строк кода, чтобы нарисовать один треугольник
- Крайне низкоуровневый API
- Последовательности команд для выполнения на GPU (command queues) в явном виде
 - Можно распараллелить генерацию command queues на несколько CPU
- Похож на DirectX 12, Metal
- vulkan-tutorial.com



- 200 строк кода, чтобы нарисовать один треугольник



- 200 строк кода, чтобы нарисовать один треугольник
- Ориентирован на web, но есть реализации под десктоп (`wgpu-native`)



- 200 строк кода, чтобы нарисовать один треугольник
- Ориентирован на web, но есть реализации под десктоп (`wgpu-native`)
- Средне низкоуровневый API: сложнее (но приятнее), чем OpenGL, и легче, чем Vulkan
- Хороший выбор для изучения после OpenGL
- sotrkh.github.io/learn-wgpu

Разновидности OpenGL

- OpenGL

Разновидности OpenGL

- OpenGL
- OpenGL ES (Embedded Systems)
 - OpenGL ES 1.0 ≈ OpenGL 1.3
 - OpenGL ES 2.0 ≈ OpenGL 2.0
 - OpenGL ES 3.0 ≈ OpenGL 3.0

Разновидности OpenGL

- OpenGL
- OpenGL ES (Embedded Systems)
 - OpenGL ES 1.0 ≈ OpenGL 1.3
 - OpenGL ES 2.0 ≈ OpenGL 2.0
 - OpenGL ES 3.0 ≈ OpenGL 3.0
- WebGL
 - WebGL 1.0 ≈ OpenGL ES 2.0
 - WebGL 2.0 ≈ OpenGL ES 3.0

Разновидности OpenGL

- OpenGL
- OpenGL ES (Embedded Systems)
 - OpenGL ES 1.0 ≈ OpenGL 1.3
 - OpenGL ES 2.0 ≈ OpenGL 2.0
 - OpenGL ES 3.0 ≈ OpenGL 3.0
- WebGL
 - WebGL 1.0 ≈ OpenGL ES 2.0
 - WebGL 2.0 ≈ OpenGL ES 3.0
- OpenGL SC (Safety Critical)
 - Убраны любые способы отстрелить себе ногу, в ущерб производительности

Что такое OpenGL?

- Не библиотека
- Спецификация API (документ) на языке C
 - Описание констант-перечислений (тэгов)
 - Описание сигнатур функций и их семантики

Что такое OpenGL?

- Не библиотека
- Спецификация API (документ) на языке C
 - Описание констант-перечислений (тэгов)
 - Описание сигнатур функций и их семантики
- OpenGL 4.6 Specification

Что такое реализация OpenGL?

Что такое реализация OpenGL?

Заголовочный файл, поставляемый системой или драйвером

- Определение типов, e.g.

```
typedef unsigned int GLenum;
```

- Определение констант, e.g.

```
#define GL_TEXTURE_2D 0x0DE1
```

- Объявление функций, e.g.

```
void glBindTexture(GLenum target, GLuint texture)
```

Что такое реализация OpenGL?

Что такое реализация OpenGL?

Бинарная реализация объявленных функций (обычно – динамическая библиотека), поставляемая системой и/или драйвером

- Может содержать непосредственную реализацию OpenGL как часть драйвера и общаться с GPU
- Может быть промежуточным звеном, маршрутизирующим вызов до драйвера
- Может быть заглушкой

Что такое реализация OpenGL?

- Заголовочный файл
 - Linux: `GL/gl.h` – до OpenGL 1.3

Что такое реализация OpenGL?

- Заголовочный файл
 - Linux: `GL/gl.h` – до OpenGL 1.3
 - Windows: `GL/gl.h` – до OpenGL 1.1

Что такое реализация OpenGL?

- Заголовочный файл
 - Linux: `GL/gl.h` – до OpenGL 1.3
 - Windows: `GL/gl.h` – до OpenGL 1.1
 - MacOS: `OpenGL/gl.h` – до OpenGL 2.1
 - Не `OpenGL/OpenGL.h`
 - Все платформы: `GL/glext.h` вместе с
`#define GL_GLEXT_PROTOTYPES` – до OpenGL 4.6

Что такое реализация OpenGL?

- Динамическая библиотека
 - Linux: `libGL.so`
 - Windows: `opengl32.dll`
 - MacOS: `OpenGL framework`

Что такое реализация OpenGL?

- Динамическая библиотека
 - Linux: `libGL.so`
 - Windows: `opengl32.dll`
 - MacOS: `OpenGL framework`
- Может не содержать функции всех версий OpenGL
 - Под Linux обычно содержит
- Остальные функции OpenGL нужно динамически загружать специфичными для платформы средствами
- ⇒ Библиотеки-загрузчики OpenGL

Загрузчики OpenGL

- C и C++ specific, для других языков обычно встроено в обёртку над OpenGL
- khronos.org/opengl/wiki/OpenGL_Loading_Library
- Обычно содержат код, автоматически сгенерированный по XML-спецификации OpenGL
- Есть мой, основанный на `glLoadGen` (который перестали поддерживать):
github.com/lisyarus/opengl-loader-generator
- Мы будем использовать [GLEW](#)

Контекст OpenGL

- Необходим для вызова любой функции OpenGL
- Привязан к конкретной реализации OpenGL
- Привязан к конкретным версии и профилю OpenGL
- Привязан к экрану / окну оконной системы / изображению в памяти
- Хранит текущее глобальное состояние OpenGL
- [khronos.org/opengl/wiki/OpenGL_Context](https://www.khronos.org/opengl/wiki/OpenGL_Context)
- Создаётся специфичными для платформы средствами
- ⇒ Библиотеки, создающие контекст OpenGL

Библиотеки, создающие контекст OpenGL

- Обычно привязывают контекст к окну и умеют обрабатывать события оконной системы
- GLUT – устаревшая, плохой интерфейс
- GLFW
- SDL2 – умеет загружать изображения, выводить звук, и другое
- `open.gl/context`

Как начать работать с OpenGL?

```
window = createWindow(title)
context = createGLContext(window, version, profile)
context.makeCurrent()
loadGLFunctions()
// тут можно работать с OpenGL!
```

Литература, ссылки

- Realtime графика
 - Computer Graphics: Principles and Practice – книжка начального уровня
 - Real-Time Rendering (4th edition) – обзор передовых алгоритмов индустрии
 - GPU Gems 1, 2, 3 – журнал про техники и алгоритмы
- OpenGL
 - khronos.org/opengl/wiki – подробное изложение всех аспектов OpenGL
 - docs.gl – удобная документация по отдельным функциям
 - learnopengl.com – уроки по отдельным темам