

Компьютерная графика

Лекция 1: Введение в курс

2021

Баллы

Баллы

- ▶ 15 баллов - работа на практиках

Баллы

- ▶ 15 баллов - работа на практиках
 - ▶ 1 балл - получилось хоть что-нибудь
 - ▶ 2 балл - получилось всё

Баллы

- ▶ 15 баллов - работа на практиках
 - ▶ 1 балл - получилось хоть что-нибудь
 - ▶ 2 балл - получилось всё
 - ▶ Можно прислать в течение 2х дней после практики (до 24:00 среды)

Баллы

- ▶ 15 баллов - работа на практиках
 - ▶ 1 балл - получилось хоть что-нибудь
 - ▶ 2 балл - получилось всё
 - ▶ Можно прислать в течение 2х дней после практики (до 24:00 среды)
- ▶ 15 баллов - каждое из 4х домашних заданий
 - ▶ Можно получить неполный балл

Баллы

- ▶ 15 баллов - работа на практиках
 - ▶ 1 балл - получилось хоть что-нибудь
 - ▶ 2 балл - получилось всё
 - ▶ Можно прислать в течение 2х дней после практики (до 24:00 среды)
- ▶ 15 баллов - каждое из 4х домашних заданий
 - ▶ Можно получить неполный балл
- ▶ Для экзамена: 25 баллов - экзамен

Баллы

- ▶ 15 баллов - работа на практиках
 - ▶ 1 балл - получилось хоть что-нибудь
 - ▶ 2 балл - получилось всё
 - ▶ Можно прислать в течение 2х дней после практики (до 24:00 среды)
- ▶ 15 баллов - каждое из 4х домашних заданий
 - ▶ Можно получить неполный балл
- ▶ Для экзамена: 25 баллов - экзамен
- ▶ Для зачёта: 25 баллов - итоговый проект

Что такое компьютерная графика?

Что такое компьютерная графика?

- ▶ Кинематограф, мультипликация

The Matrix Revolutions (2003)



Avatar (2009)



The Avengers (2012)



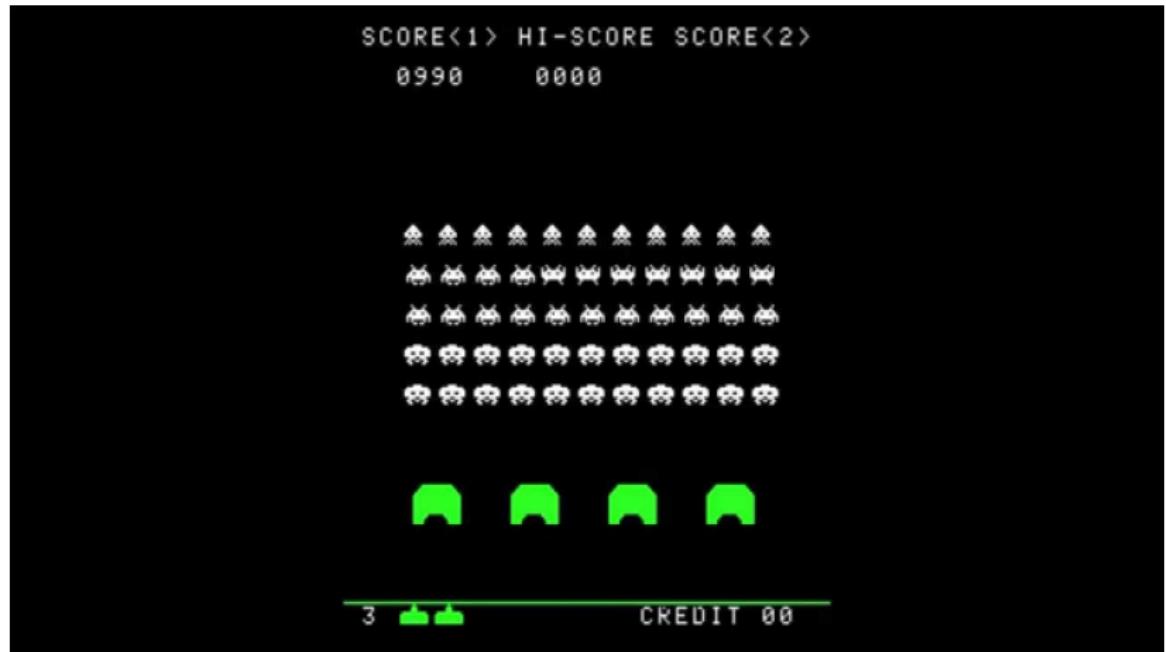
Klaus (2019)



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)



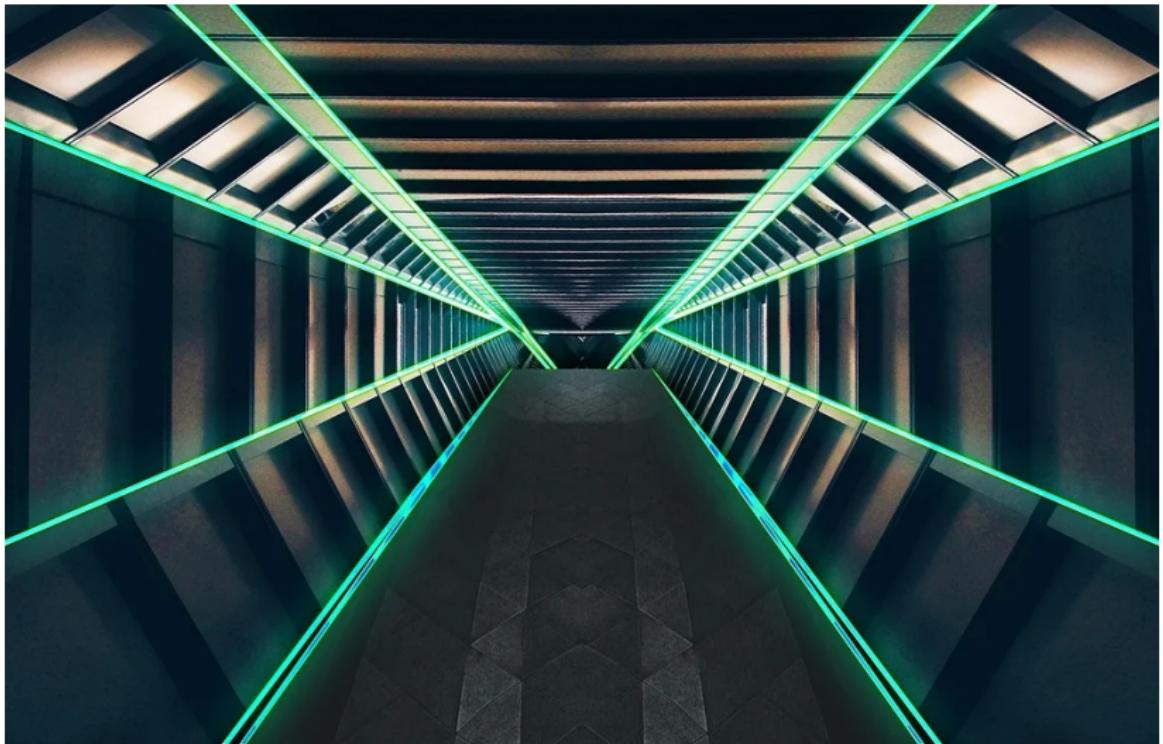
Cyberpunk 2077 (2020)



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

Toggle switch	Off	Disabled Off
<input type="button" value="Off"/>	<input type="button" value="Disabled Off"/>	<input type="button" value=""/>
<input type="button" value="On"/>	<input type="button" value="Disabled On"/>	<input type="button" value=""/>

Europa Universalis 4

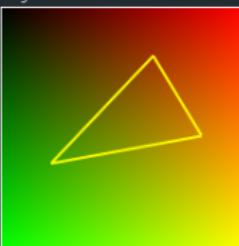


Dear ImGui

▼ Example: Custom rendering

Clear Undo

Left-click and drag to add
Right-click to undo



E028E4: 00 00 F8 44 00 40 7E 44 ...D, @-D
E028EC: 06 33 88 3C 00 00 A0 40 ...3,<...@
E028F4: 18 BA DE 00 08 BA DE 00
E028FC: 9A 99 99 3E 00 00 C0 40 ...>..@
E02984: 00 00 C0 40 02 01 00 00 ...@...
E0298C: 07 01 00 00 06 01 00 00
E02914: 09 01 00 00 08 01 00 00
E0291C: 0A 01 00 00 08 01 00 00
E02924: 0C 01 00 00 0D 01 00 00
E0292C: 05 01 00 00 03 01 00 00
E02934: 01 01 00 00 08 01 00 00
E0293C: 41 00 00 00 43 00 00 00 A...C...
E02944: 56 00 00 00 58 00 00 00 E...X...Console
E0294C: 59 00 00 00 5A 00 00 00 Y...Z...
E02954: 00 00 88 3E CD CC 4C 3D #CLASSIFY implements a console with basic coloring,
E0295C: 00 00 00 00 A4 28 E0 80 #HELP, and history. A more elaborate implementation may
support more console entries along with extra data such as timestamp,
etc.

8 rows Range E028E4..E03C93 Iter. etc.

▼ Example: Layout

File

MyObject 0	MyObject 1	MyObject 2	MyObject 3	MyObject 4	MyObject 5	MyObject 6	MyObject 7	MyObject 8	MyObject 9	MyObject 10	MyObject 11	MyObject 12	MyObject 13
------------	------------	------------	------------	------------	------------	------------	------------	------------	------------	-------------	-------------	-------------	-------------

Revert Save

Enter 'HELP' for help, press TAB to use text completion.

Add Dummy Text Add Dummy Error Clear

Filter ("incl,-excl") ("error")

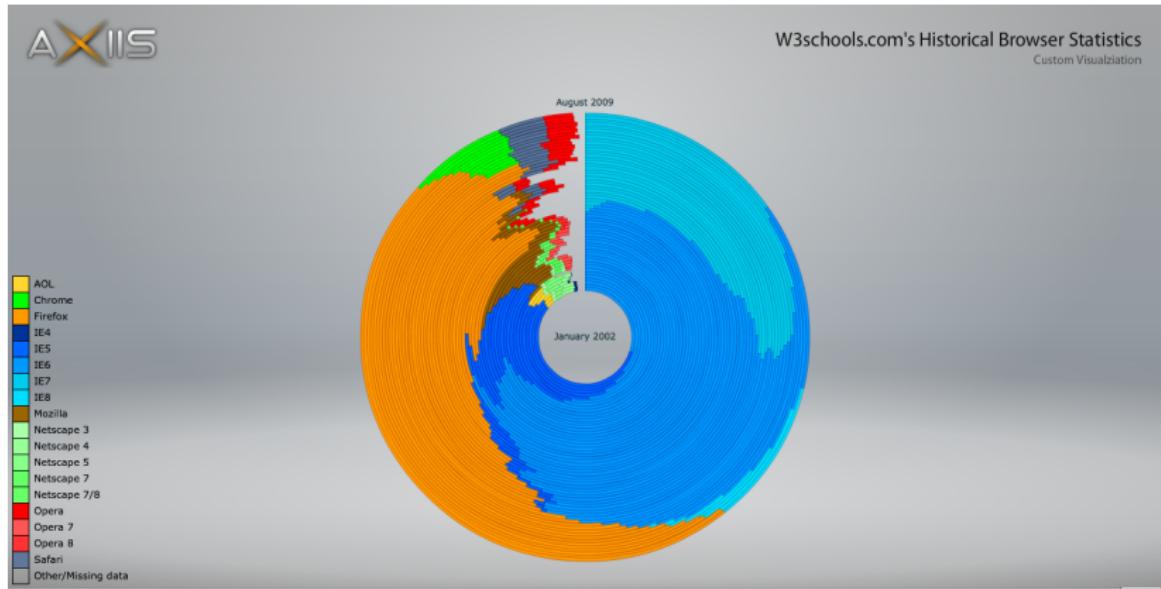
0 some text
some more text
display very important message here!
[error] something went wrong
Possible matches:
- HELP
- HISTORY
HELP
Commands:
- HELP
- HISTORY
- CLEAR
- CLASSIFY
hello, imgui world!
Unknown command: 'hello, imgui world!'

hello, imgui world! Input

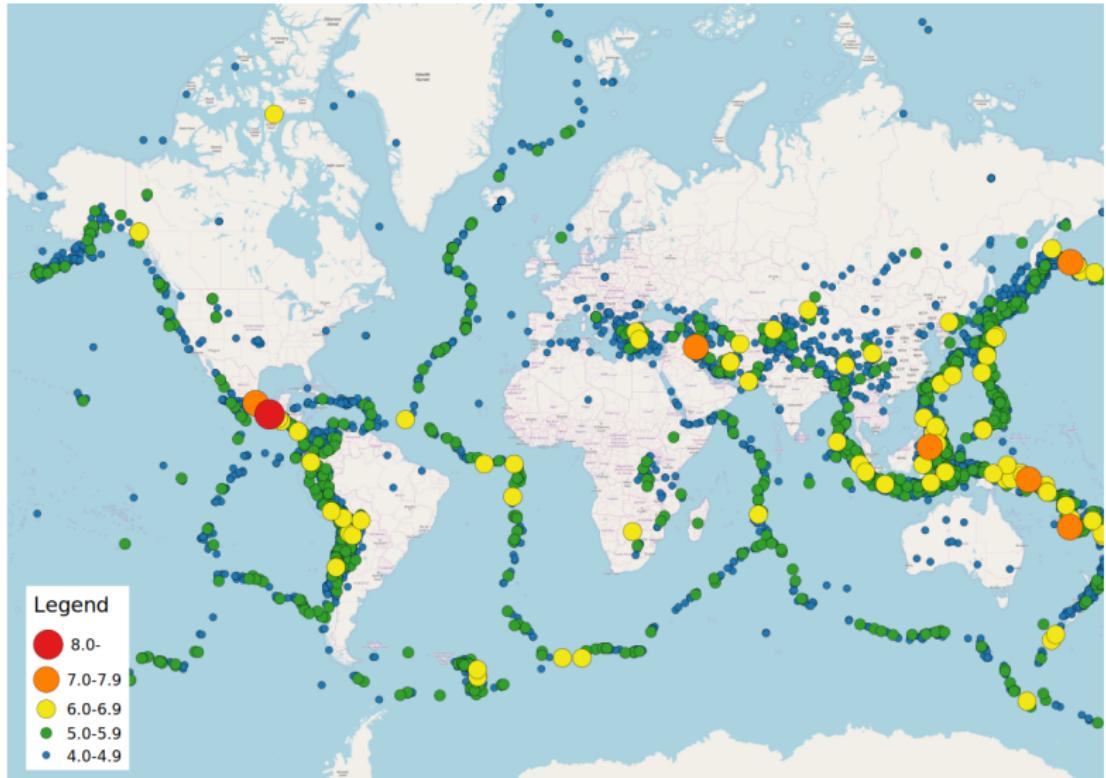
Что такое компьютерная графика?

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

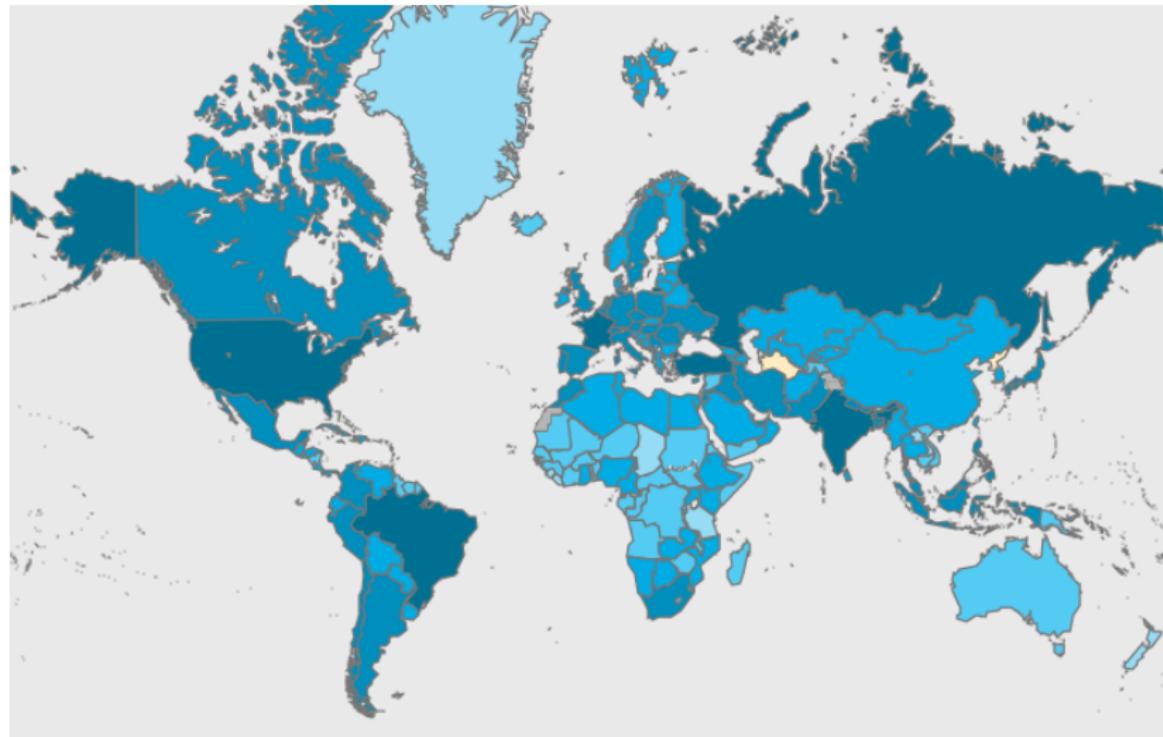
Популярность браузеров в 2002-2009



Карта землетрясений



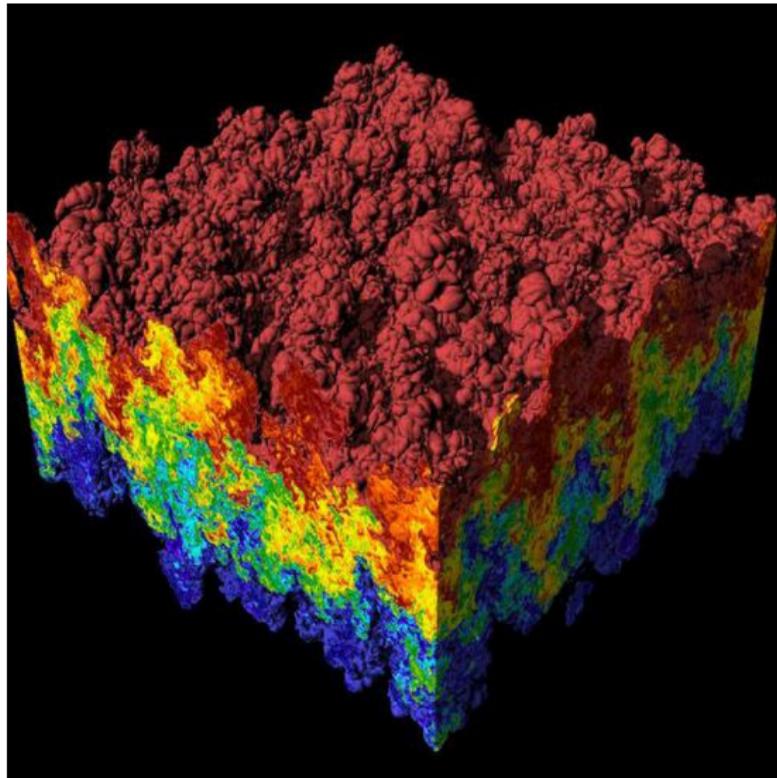
Количество случаев заражения COVID-19



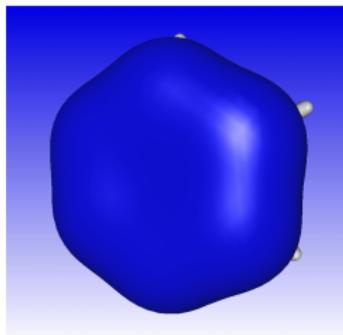
Что такое компьютерная графика?

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

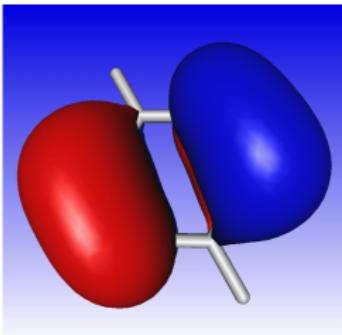
Неустойчивость Рэлея — Тейлора



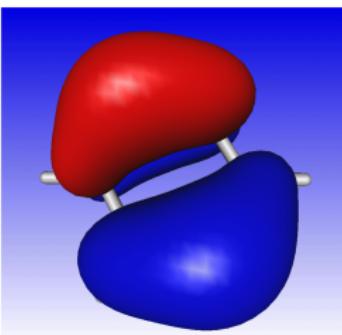
Молекулярные орбитали бензола



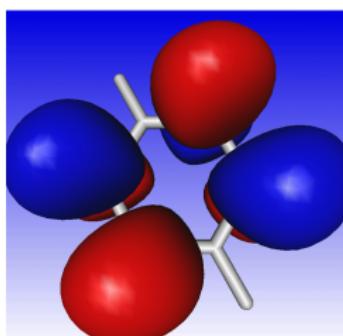
16



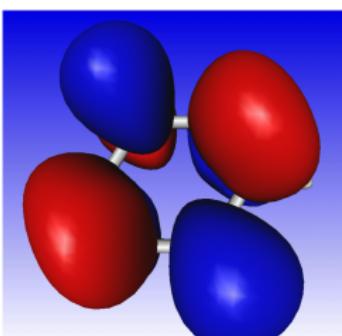
20



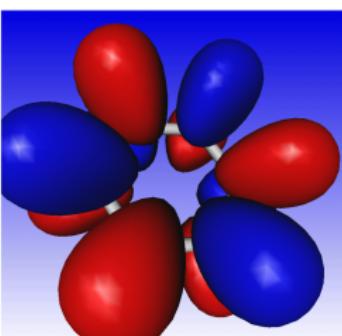
21



22

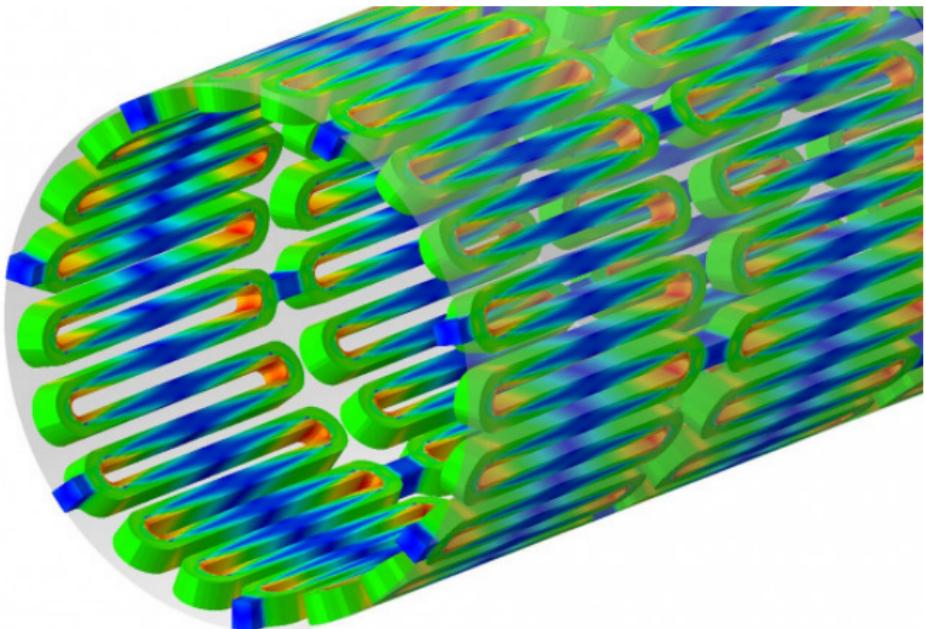


23



30

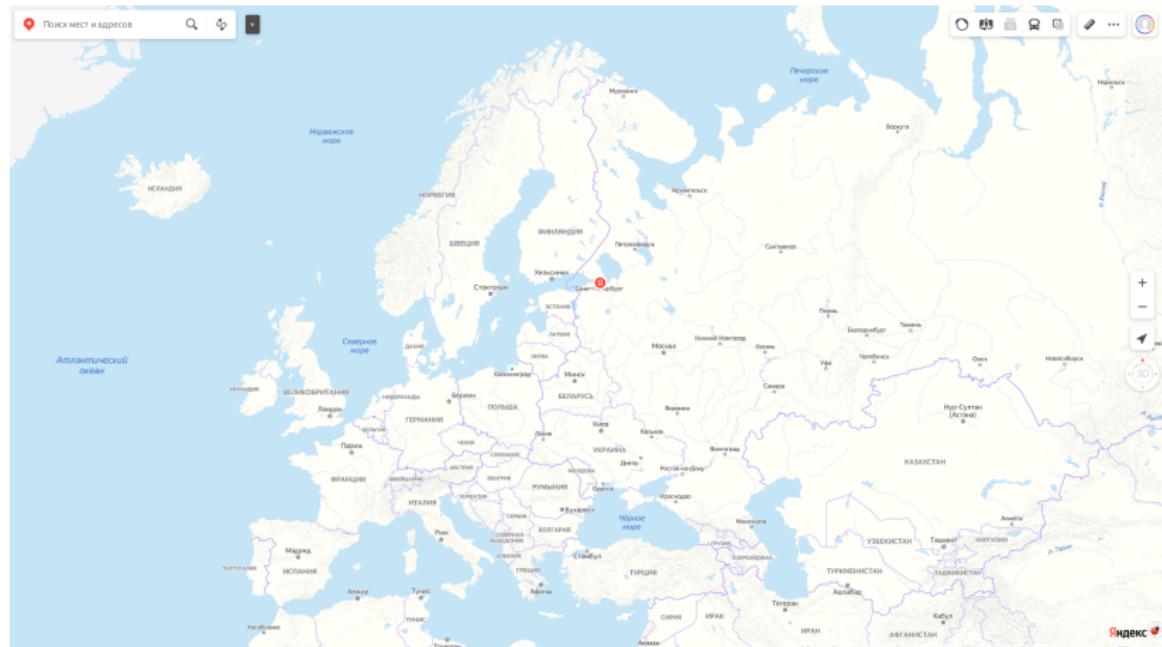
Симуляция напряжений в стене методом конечных элементов



Что такое компьютерная графика?

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

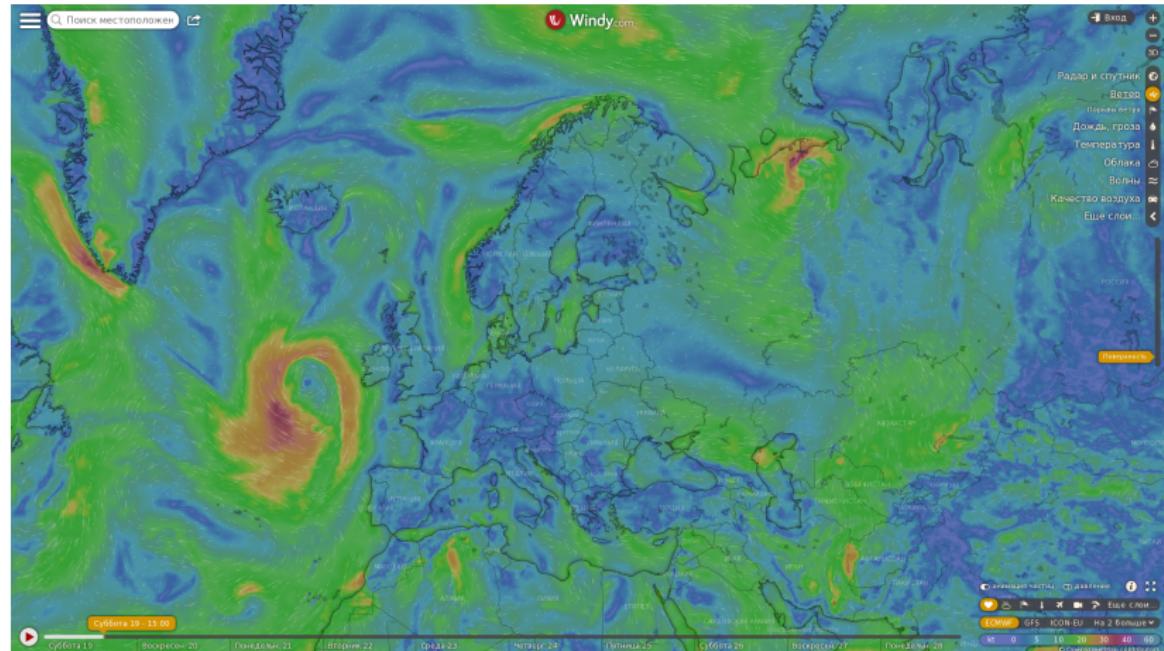
Схематическая карта



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

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 реализовывать графические движки

Чем мы будем заниматься?

- ▶ Как реализовывать графические движки
- ▶ Как реализовывать графические эффекты

Чем мы будем заниматься?

- ▶ Как реализовывать графические движки
- ▶ Как реализовывать графические эффекты
- ▶ Как их оптимизировать

Чем мы будем заниматься?

- ▶ Как реализовывать графические движки
- ▶ Как реализовывать графические эффекты
- ▶ Как их оптимизировать
- ▶ Graphics engineer

Где это пригодится?

Где это пригодится?

- ▶ Разработка игр

Где это пригодится?

- ▶ Разработка игр
- ▶ Разработка инструментов для художников/дизайнеров/архитекторов/etc

Где это пригодится?

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

Где это пригодится?

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

Где это пригодится?

- ▶ Разработка игр
- ▶ Разработка инструментов для художников/дизайнеров/архитекторов/etc
- ▶ Разработка инструментов для научной визуализации/визуализации данных/etc
- ▶ Разработка картографических приложений
- ▶ И т.д.

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 (1980e - 1990e)
- ▶ OpenGL (Silicon Graphics, 1992)

Как использовать GPU? Графические API:

GPU - Graphics Processing Unit

- ▶ Вендор-специфичные API (1980e - 1990e)
- ▶ 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, в разработке)

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, в разработке)

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, в разработке)

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)

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
- ▶ Низкий порог входления

Почему OpenGL 3.3?

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

История графических API: OpenGL 1.0

```
for o in scene.objects:  
    glBegin(GL_TRIANGLES)  
        for t in o.triangles:  
            for v in t.vertices:  
                glColor3f(v.color)  
                glNormal3f(v.normal)  
                glVertex3f(v.position)  
    glEnd(GL_TRIANGLES)
```

История графических API: OpenGL 1.0

```
for o in scene.objects:  
    glBegin(GL_TRIANGLES)  
        for t in o.triangles:  
            for v in t.vertices:  
                glColor3f(v.color)  
                glNormal3f(v.normal)  
                glVertex3f(v.position)  
    glEnd(GL_TRIANGLES)
```

- ▶ Данные хранятся в памяти CPU

История графических API: OpenGL 1.0

```
for o in scene.objects:  
    glBegin(GL_TRIANGLES)  
        for t in o.triangles:  
            for v in t.vertices:  
                glColor3f(v.color)  
                glNormal3f(v.normal)  
                glVertex3f(v.position)  
    glEnd(GL_TRIANGLES)
```

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

История графических API: OpenGL 1.0

```
for o in scene.objects:  
    glBegin(GL_TRIANGLES)  
        for t in o.triangles:  
            for v in t.vertices:  
                glColor3f(v.color)  
                glNormal3f(v.normal)  
                glVertex3f(v.position)  
    glEnd(GL_TRIANGLES)
```

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

История графических API: OpenGL 1.0 (1992)

- ▶ Предварительно записать список команд в отдельную сущность - *display list*
 - ▶ Выполнить все команды из списка одной командой `glCallList`
 - ▶ Не любой OpenGL-вызов может быть частью *display list*'а
 - ▶ Предок *command queues* в Vulkan и подобных API

История графических API: OpenGL 1.0 (1992)

- ▶ Вершины задаются OpenGL-вызовами
- ▶ Display lists
- ▶ Матрицы преобразований вершин: позиционирование объектов и камеры
- ▶ Fixed-function pipeline: настраиваемая, но не расширяемая последовательность операций
- ▶ Асинхронный API: команды выполняются на GPU когда-нибудь

История графических API: OpenGL 1.1 (1997)

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

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

- ▶ В текстурах можно записать очень много интересного: normal map, material map, bump map
- ▶ Хочется выполнять сложные вычисления на каждый пиксель
- ▶ ⇒ Texture environments - зачатки программируемости GPU

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
- ▶ Occlusion query - асинхронный механизм узнать, был ли нарисован объект

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

История графических API: OpenGL 2.0 (2004)

```
// на старте
for o in scene.objects:
    o.createShader()
    o.uploadVertices()

// при рендеринге
for o in scene.objects:
    glUseProgram(o.shader)
    glBindVertexArray(o.vertexArray)
    glDrawArrays(o.vertexCount)
```

История графических API: OpenGL 3.0 (2008)

- ▶ Огромная часть API объявлена deprecated

История графических API: OpenGL 3.0 (2008)

- ▶ Огромная часть API объявлена deprecated
- ▶ 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
- ▶ Геометрические шейдеры - возможность менять тип геометрии и количество вершин на лету

История графических API: OpenGL 3.3 (2010)

- ▶ Улучшенный *instanced rendering*
 - ▶ Часть вершинных атрибутов может быть общей для всех копий, другая часть меняться от копии к копии

История графических API: OpenGL 3.3 (2010)

- ▶ Улучшенный *instanced rendering*
 - ▶ Часть вершинных атрибутов может быть общей для всех копий, другая часть меняться от копии к копии
- ▶ Texture swizzle

История графических API: OpenGL 3.3 (2010)

- ▶ Улучшенный *instanced rendering*
 - ▶ Часть вершинных атрибутов может быть общей для всех копий, другая часть меняться от копии к копии
- ▶ Texture swizzle
- ▶ Нумерация версий языка шейдеров GLSL синхронизирована с версиями OpenGL

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

История графических API: Vulkan 1.1 (2018)

- ▶ 700 строк кода, чтобы нарисовать один треугольник

История графических API: Vulkan 1.1 (2018)

- ▶ 700 строк кода, чтобы нарисовать один треугольник
- ▶ Крайне низкоуровневый API
- ▶ Последовательности команд (command queues) в явном виде (display lists done right)
 - ▶ Можно распараллелить генерацию command queues на несколько CPU

История графических API: Vulkan 1.1 (2018)

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

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, но 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 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?

- ▶ 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);`
- ▶ 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 вызовы 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

- ▶ С 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
- ▶ Мы будем использовать [GLEW](#)

Контекст OpenGL

- ▶ Привязан к конкретной реализации OpenGL
- ▶ Привязан к конкретной версии OpenGL
- ▶ Привязан к экрану / окну оконной системы / изображению в памяти
- ▶ Хранит текущее глобальное состояние OpenGL
- ▶ 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