

# Компьютерная графика

Лекция 13: состояние OpenGL (напоминание), матрицы проекций (напоминание), рендеринг в subwindow, дистрибуция приложений на OpenGL

2021

# Настройка графического конвейера

- ▶ Графический конвейер (pipeline) - набор всех операций, происходящих с данными от момента вызова `glDraw*` до появления пикселей на экране (или текстуре/рендербуфере)

# Настройка графического конвейера

- ▶ Графический конвейер (pipeline) - набор всех операций, происходящих с данными от момента вызова `glDraw*` до появления пикселей на экране (или текстуре/рендербуфере)
- ▶ Графический конвейер = programmable pipeline + fixed-function pipeline

# Настройка графического конвейера

- ▶ Графический конвейер (pipeline) - набор всех операций, происходящих с данными от момента вызова `glDraw*` до появления пикселей на экране (или текстуре/рендербуфере)
- ▶ Графический конвейер = programmable pipeline + fixed-function pipeline
- ▶ Настройка programmable pipeline: шейдеры (шейдерная программа)

# Настройка графического конвейера

- ▶ Графический конвейер (pipeline) - набор всех операций, происходящих с данными от момента вызова `glDraw*` до появления пикселей на экране (или текстуре/рендербуфере)
- ▶ Графический конвейер = programmable pipeline + fixed-function pipeline
- ▶ Настройка programmable pipeline: шейдеры (шейдерная программа)
- ▶ Настройка fixed-function pipeline: включение/выключение (`glEnable/glDisable`) конкретных операций и их специфическая настройка

# Настройка fixed-function pipeline

- ▶ Depth clamp
  - ▶ По умолчанию, все примитивы обрезаются по уравнению  $z \leq |w|$
  - ▶ Можно заменить обрезание clamping'ом через `glEnable(GL_DEPTH_CLAMP)`

# Настройка fixed-function pipeline

- ▶ Depth clamp
  - ▶ По умолчанию, все примитивы обрезаются по уравнению  $z \leq |w|$
  - ▶ Можно заменить обрезание clamping'ом через `glEnable(GL_DEPTH_CLAMP)`
- ▶ Culling
  - ▶ Можно не рисовать back-facing или front-facing полигоны
  - ▶ Включить: `glEnable(GL_CULL_FACE)`
  - ▶ Настроить, что **не** рисуется: `glCullFace`
  - ▶ Настроить, что считается back-facing, а что front-facing: `glFrontFace`

# Настройка fixed-function pipeline

- ▶ Depth clamp
  - ▶ По умолчанию, все примитивы обрезаются по уравнению  $z \leq |w|$
  - ▶ Можно заменить обрезание clamping'ом через `glEnable(GL_DEPTH_CLAMP)`
- ▶ Culling
  - ▶ Можно не рисовать back-facing или front-facing полигоны
  - ▶ Включить: `glEnable(GL_CULL_FACE)`
  - ▶ Настроить, что **не** рисуется: `glCullFace`
  - ▶ Настроить, что считается back-facing, а что front-facing: `glFrontFace`
- ▶ Viewport
  - ▶ Настроить перевод из NDC (normalized device coordinates, [-1..1]) в пиксельные координаты: `glViewport`
  - ▶ Обычно нужно делать каждый раз при изменении размеров окна или при переключении фреймбуферов



# Настройка fixed-function pipeline

## ▶ Depth test

- ▶ Можно не рисовать пиксели, находящиеся сзади уже нарисованных пикселей
- ▶ Включить: `glEnable(GL_DEPTH_TEST)`
- ▶ Настроить: `glDepthFunc`
- ▶ Настроить преобразование из NDC в  $[0, 1]$ : `glDepthRangef`
- ▶ Включить/выключить запись значений глубины: `glDepthMask`

# Настройка fixed-function pipeline

## ▶ Depth test

- ▶ Можно не рисовать пиксели, находящиеся сзади уже нарисованных пикселей
- ▶ Включить: `glEnable(GL_DEPTH_TEST)`
- ▶ Настроить: `glDepthFunc`
- ▶ Настроить преобразование из NDC в  $[0, 1]$ : `glDepthRange`
- ▶ Включить/выключить запись значений глубины: `glDepthMask`

## ▶ Stencil test

- ▶ Включить: `glEnable(GL_STENCIL_TEST)`
- ▶ Настроить: `glStencilFunc`, `glStencilOp`, `glStencilMask`

# Настройка fixed-function pipeline

## ▶ Depth test

- ▶ Можно не рисовать пиксели, находящиеся сзади уже нарисованных пикселей
- ▶ Включить: `glEnable(GL_DEPTH_TEST)`
- ▶ Настроить: `glDepthFunc`
- ▶ Настроить преобразование из NDC в  $[0, 1]$ : `glDepthRangef`
- ▶ Включить/выключить запись значений глубины: `glDepthMask`

## ▶ Stencil test

- ▶ Включить: `glEnable(GL_STENCIL_TEST)`
- ▶ Настроить: `glStencilFunc`, `glStencilOp`, `glStencilMask`

## ▶ Scissor test

- ▶ Можно не рисовать пиксели, находящиеся вне некоторого прямоугольника
- ▶ Включить: `glEnable(GL_SCISSOR_TEST)`
- ▶ Настроить: `glScissor`

# Настройка fixed-function pipeline

- ▶ Color mask
  - ▶ Настроить запись в конкретные цветовые каналы:  
`glColorMask`

# Настройка fixed-function pipeline

- ▶ Color mask
  - ▶ Настроить запись в конкретные цветовые каналы:  
`glColorMask`
- ▶ Blending
  - ▶ Можно записывать значение некоторой функции от входного цвета и уже записанного цвета
  - ▶ Включить: `glEnable(GL_BLEND)`
  - ▶ Настроить: `glBlendFunc/glBlendFuncSeparate`,  
`glBlendEquation`, `glBlendColor`

# Настройка fixed-function pipeline

- ▶ Color mask
  - ▶ Настроить запись в конкретные цветовые каналы: `glColorMask`
- ▶ Blending
  - ▶ Можно записывать значение некоторой функции от входного цвета и уже записанного цвета
  - ▶ Включить: `glEnable(GL_BLEND)`
  - ▶ Настроить: `glBlendFunc/glBlendFuncSeparate`, `glBlendEquation`, `glBlendColor`
- ▶ Color logical operation
  - ▶ Можно записывать результат некоторой побитовой операции от входного цвета и уже записанного цвета
  - ▶ Выключает blending
  - ▶ Включить: `glEnable(GL_COLOR_LOGIC_OP)`
  - ▶ Настроить: `glLogicOp`

# Проекции

- ▶ Ничто не мешает делать с вершинами абсолютно любые преобразования в вершинном шейдере

# Проекции

- ▶ Ничто не мешает делать с вершинами абсолютно любые преобразования в вершинном шейдере
- ▶ Обычно используют ортографическую или перспективную проекцию, так как они выражаются матрицами (с учётом perspective divide)



# Проекции

- ▶ Ничто не мешает делать с вершинами абсолютно любые преобразования в вершинном шейдере
- ▶ Обычно используют ортографическую или перспективную проекцию, так как они выражаются матрицами (с учётом perspective divide)
- ▶ Ортографическая проекция: **не использует** perspective divide, последняя строка равна  $(0\ 0\ 0\ 1)$

# Проекции

- ▶ Ничто не мешает делать с вершинами абсолютно любые преобразования в вершинном шейдере
- ▶ Обычно используют ортографическую или перспективную проекцию, так как они выражаются матрицами (с учётом perspective divide)
- ▶ Ортографическая проекция: **не использует** perspective divide, последняя строка равна  $(0\ 0\ 0\ 1)$
- ▶ Перспективная проекция: **использует** perspective divide, последняя строка содержит зависимость от  $X$ ,  $Y$  или  $Z$

# Ортографическая проекция

- ▶ Проецирует параллельно некому вектору: точки пространства, отличающиеся на вектор, параллельный направлению взгляда, проецируются в одну точку экрана

# Ортографическая проекция

- ▶ Проецирует параллельно некому вектору: точки пространства, отличающиеся на вектор, параллельный направлению взгляда, проецируются в одну точку экрана
- ▶ Видимая область пространства - параллелепипед

# Ортографическая проекция

- ▶ Проецирует параллельно некому вектору: точки пространства, отличающиеся на вектор, параллельный направлению взгляда, проецируются в одну точку экрана
- ▶ Видимая область пространства - параллелепипед
- ▶ Видимый размер объектов **не зависит** от расстояния до них

# Ортографическая проекция

- ▶ Проецирует параллельно некому вектору: точки пространства, отличающиеся на вектор, параллельный направлению взгляда, проецируются в одну точку экрана
- ▶ Видимая область пространства - параллелепипед
- ▶ Видимый размер объектов **не зависит** от расстояния до них
- ▶ Удобно описать центром видимой области  $C$  и осями  $X, Y, Z$

# Ортографическая проекция

- ▶ Проецирует параллельно некому вектору: точки пространства, отличающиеся на вектор, параллельный направлению взгляда, проецируются в одну точку экрана
- ▶ Видимая область пространства - параллелепипед
- ▶ Видимый размер объектов **не зависит** от расстояния до них
- ▶ Удобно описать центром видимой области  $C$  и осями  $X, Y, Z$ 
  - ▶ Центр  $C$  переходит в центр экрана  $((0, 0, 0)$  в NDC)

# Ортографическая проекция

- ▶ Проецирует параллельно некому вектору: точки пространства, отличающиеся на вектор, параллельный направлению взгляда, проецируются в одну точку экрана
- ▶ Видимая область пространства - параллелепипед
- ▶ Видимый размер объектов **не зависит** от расстояния до них
- ▶ Удобно описать центром видимой области  $C$  и осями  $X, Y, Z$ 
  - ▶ Центр  $C$  переходит в центр экрана  $((0, 0, 0)$  в NDC)
  - ▶ Точки, отличающиеся на вектор параллельный  $X$ , после проекции отличаются только  $X$ -координатой



# Ортографическая проекция

- ▶ Проецирует параллельно некому вектору: точки пространства, отличающиеся на вектор, параллельный направлению взгляда, проецируются в одну точку экрана
- ▶ Видимая область пространства - параллелепипед
- ▶ Видимый размер объектов **не зависит** от расстояния до них
- ▶ Удобно описать центром видимой области  $C$  и осями  $X, Y, Z$ 
  - ▶ Центр  $C$  переходит в центр экрана  $((0, 0, 0)$  в NDC)
  - ▶ Точки, отличающиеся на вектор параллельный  $X$ , после проекции отличаются только  $X$ -координатой
  - ▶ Точки, отличающиеся на вектор параллельный  $Y$ , после проекции отличаются только  $Y$ -координатой

# Ортографическая проекция

- ▶ Проецирует параллельно некому вектору: точки пространства, отличающиеся на вектор, параллельный направлению взгляда, проецируются в одну точку экрана
- ▶ Видимая область пространства - параллелепипед
- ▶ Видимый размер объектов **не зависит** от расстояния до них
- ▶ Удобно описать центром видимой области  $C$  и осями  $X, Y, Z$ 
  - ▶ Центр  $C$  переходит в центр экрана ( $(0, 0, 0)$  в NDC)
  - ▶ Точки, отличающиеся на вектор параллельный  $X$ , после проекции отличаются только  $X$ -координатой
  - ▶ Точки, отличающиеся на вектор параллельный  $Y$ , после проекции отличаются только  $Y$ -координатой
  - ▶ Точки, отличающиеся на вектор параллельный  $Z$ , после проекции попадают в один пиксель (и отличаются только  $Z$ -координатой)

# Ортографическая проекция

- ▶ Проецирует параллельно некому вектору: точки пространства, отличающиеся на вектор, параллельный направлению взгляда, проецируются в одну точку экрана
- ▶ Видимая область пространства - параллелепипед
- ▶ Видимый размер объектов **не зависит** от расстояния до них
- ▶ Удобно описать центром видимой области  $C$  и осями  $X, Y, Z$ 
  - ▶ Центр  $C$  переходит в центр экрана  $((0, 0, 0)$  в NDC)
  - ▶ Точки, отличающиеся на вектор параллельный  $X$ , после проекции отличаются только  $X$ -координатой
  - ▶ Точки, отличающиеся на вектор параллельный  $Y$ , после проекции отличаются только  $Y$ -координатой
  - ▶ Точки, отличающиеся на вектор параллельный  $Z$ , после проекции попадают в один пиксель (и отличаются только  $Z$ -координатой)
- ▶ N.B.: можно понимать её как композицию аффинного преобразования,двигающего камеру в точку  $C$  с осями  $XYZ$ , и стандартной ортографической проекции (с единичной матрицей)

# Ортографическая проекция

- ▶ Проецирует параллельно некому вектору: точки пространства, отличающиеся на вектор, параллельный направлению взгляда, проецируются в одну точку экрана
- ▶ Видимая область пространства - параллелепипед
- ▶ Видимый размер объектов **не зависит** от расстояния до них
- ▶ Удобно описать центром видимой области  $C$  и осями  $X, Y, Z$ 
  - ▶ Центр  $C$  переходит в центр экрана ( $(0, 0, 0)$  в NDC)
  - ▶ Точки, отличающиеся на вектор параллельный  $X$ , после проекции отличаются только  $X$ -координатой
  - ▶ Точки, отличающиеся на вектор параллельный  $Y$ , после проекции отличаются только  $Y$ -координатой
  - ▶ Точки, отличающиеся на вектор параллельный  $Z$ , после проекции попадают в один пиксель (и отличаются только  $Z$ -координатой)
- ▶ N.B.: можно понимать её как композицию аффинного преобразования,двигающего камеру в точку  $C$  с осями  $XYZ$ , и стандартной ортографической проекции (с единичной матрицей)
- ▶ Матрица проекции: см. лекцию 4

# Перспективная проекция

- ▶ Проецирует из некой точки: точки пространства, лежащие на одной прямой с центром проекции, проецируются в одну точку экрана

## Перспективная проекция

- ▶ Проецирует из некой точки: точки пространства, лежащие на одной прямой с центром проекции, проецируются в одну точку экрана
- ▶ Видимая область пространства - усечённая пирамида

## Перспективная проекция

- ▶ Проецирует из некой точки: точки пространства, лежащие на одной прямой с центром проекции, проецируются в одну точку экрана
- ▶ Видимая область пространства - усечённая пирамида
- ▶ Видимый размер объектов **зависит** от расстояния до них: чем объект дальше, тем он мельче

## Перспективная проекция

- ▶ Проецирует из некой точки: точки пространства, лежащие на одной прямой с центром проекции, проецируются в одну точку экрана
- ▶ Видимая область пространства - усечённая пирамида
- ▶ Видимый размер объектов **зависит** от расстояния до них: чем объект дальше, тем он мельче
- ▶ Удобно описать параметрами проекции `near`, `far`, `fov_x`, `fov_y` и аффинными преобразованием,двигающим камеру (как с ортографической проекцией)



# Перспективная проекция

- ▶ Проецирует из некой точки: точки пространства, лежащие на одной прямой с центром проекции, проецируются в одну точку экрана
- ▶ Видимая область пространства - усечённая пирамида
- ▶ Видимый размер объектов **зависит** от расстояния до них: чем объект дальше, тем он мельче
- ▶ Удобно описать параметрами проекции  $near$ ,  $far$ ,  $fov_x$ ,  $fov_y$  и аффинными преобразованием,двигающим камеру (как с ортографической проекцией)
- ▶ Обычно за направление взгляда берут ось  $-Z$ , чтобы получить левую систему координат

# Перспективная проекция

- ▶ Проецирует из некой точки: точки пространства, лежащие на одной прямой с центром проекции, проецируются в одну точку экрана
- ▶ Видимая область пространства - усечённая пирамида
- ▶ Видимый размер объектов **зависит** от расстояния до них: чем объект дальше, тем он мельче
- ▶ Удобно описать параметрами проекции  $near$ ,  $far$ ,  $fov_x$ ,  $fov_y$  и аффинными преобразованием,двигающим камеру (как с ортографической проекцией)
- ▶ Обычно за направление взгляда берут ось  $-Z$ , чтобы получить левую систему координат
  - ▶  $near$ : всё ближе этого значения (по  $Z$ -координате) будет отсекается

# Перспективная проекция

- ▶ Проецирует из некой точки: точки пространства, лежащие на одной прямой с центром проекции, проецируются в одну точку экрана
- ▶ Видимая область пространства - усечённая пирамида
- ▶ Видимый размер объектов **зависит** от расстояния до них: чем объект дальше, тем он мельче
- ▶ Удобно описать параметрами проекции  $near$ ,  $far$ ,  $fov_x$ ,  $fov_y$  и аффинными преобразованием,двигающим камеру (как с ортографической проекцией)
- ▶ Обычно за направление взгляда берут ось  $-Z$ , чтобы получить левую систему координат
  - ▶  $near$ : всё ближе этого значения (по  $Z$ -координате) будет отсекается
  - ▶  $far$ : всё дальше этого значения (по  $Z$ -координате) будет отсекается

## Перспективная проекция

- ▶ Проецирует из некой точки: точки пространства, лежащие на одной прямой с центром проекции, проецируются в одну точку экрана
- ▶ Видимая область пространства - усечённая пирамида
- ▶ Видимый размер объектов **зависит** от расстояния до них: чем объект дальше, тем он мельче
- ▶ Удобно описать параметрами проекции `near`, `far`, `fovx`, `fovy` и аффинными преобразованием,двигающим камеру (как с ортографической проекцией)
- ▶ Обычно за направление взгляда берут ось `-Z`, чтобы получить левую систему координат
  - ▶ `near`: всё ближе этого значения (по `Z`-координате) будет отсекается
  - ▶ `far`: всё дальше этого значения (по `Z`-координате) будет отсекается
  - ▶ `fovx`, `fovy` - угол обзора по `X` и `Y`

# Перспективная проекция

- ▶ Проецирует из некой точки: точки пространства, лежащие на одной прямой с центром проекции, проецируются в одну точку экрана
- ▶ Видимая область пространства - усечённая пирамида
- ▶ Видимый размер объектов **зависит** от расстояния до них: чем объект дальше, тем он мельче
- ▶ Удобно описать параметрами проекции `near`, `far`, `fovx`, `fovy` и аффинными преобразованием,двигающим камеру (как с ортографической проекцией)
- ▶ Обычно за направление взгляда берут ось `-Z`, чтобы получить левую систему координат
  - ▶ `near`: всё ближе этого значения (по `Z`-координате) будет отсекается
  - ▶ `far`: всё дальше этого значения (по `Z`-координате) будет отсекается
  - ▶ `fovx`, `fovy` - угол обзора по `X` и `Y`
- ▶ Матрица проекции: см. лекцию 4