Компьютерная графика

Лекция 12: Анимации, easing functions, bitmap-анимации, кватернионы, иерархии объектов, скелетная анимация

2021

Анимации

- Анимация (в общем смысле) – что угодно, меняющееся со временем

Анимации

- Анимация (в общем смысле) – что угодно, меняющееся со временем
  - Двигающийся объект
  - Анимированный элемент интерфейса
  - Анимированная модель
  - Движущаяся камера
  - etc.

Анимации

- Анимация (в общем смысле) – что угодно, меняющееся со временем
  - Двигающийся объект
  - Анимированный элемент интерфейса
  - Анимированная модель
  - Движущаяся камера
  - etc.
- Сводится к вопросу о том, что и как мы меняем в зависимости от времени

Анимации: общие идеи

- Вычисление анимаций лучше привязывать к реальному времени, а не к номеру кадра (`state += speed * dt` а не `state += speed`)

Анимации: общие идеи

- Вычисление анимаций лучше привязывать к реальному времени, а не к номеру кадра (`state += speed * dt` а не `state += speed`)
  - Будет лучше работать при выключенном VSync
  - Будет лучше работать, когда CPU/GPU не справляются с нагрузкой

Анимации: общие идеи

- Вычисление анимаций лучше привязывать к реальному времени, а не к номеру кадра (`state += speed * dt` а не `state += speed`)
  - Будет лучше работать при выключенном VSync
  - Будет лучше работать, когда CPU/GPU не справляются с нагрузкой
- Дискретные движения (напр. поворот камеры) можно сглаживать: вместо `x = x_target` делать `x = lerp(x, x_target, exp(-speed * dt))` (`lerp(x0,x1,t)` - функция линейной интерполяции, в GLSL она называется `mix`)

Анимации: общие идеи

- Вычисление анимаций лучше привязывать к реальному времени, а не к номеру кадра (`state += speed * dt` а не `state += speed`)
  - Будет лучше работать при выключенном VSync
  - Будет лучше работать, когда CPU/GPU не справляются с нагрузкой
- Дискретные движения (напр. поворот камеры) можно сглаживать: вместо `x = x_target` делать `x = lerp(x, x_target, exp(-speed * dt))` (`lerp(x0,x1,t)` - функция линейной интерполяции, в GLSL она называется `mix`)
  - Это соответствует численному решению уравнения $\dot{x} = speed \cdot (x_{target} - x)$
  - Не зависит от начального значения `x`

Анимации: easing functions

- При временно́й интерполяции между двумя значениями вместо линейной интерполяции
  `x = lerp(x_start, x_end, t)` можно использовать т.н. easing functions

Анимации: easing functions

- При временно́й интерполяции между двумя значениями вместо линейной интерполяции
  `x = lerp(x_start, x_end, t)` можно использовать т.н. easing functions
- Применяются к параметру интерполяции $t \in [0, 1]$ и сглаживают анимацию:
  `x = lerp(x_start, x_end, easing(t))`

Анимации: easing functions

- При временно́й интерполяции между двумя значениями вместо линейной интерполяции
  `x = lerp(x_start, x_end, t)` можно использовать т.н. easing functions
- Применяются к параметру интерполяции $t \in [0, 1]$ и сглаживают анимацию:
  `x = lerp(x_start, x_end, easing(t))`
- Примеры easing functions:
  - $f(t) = t$
  - $f(t) = 3t^2 - 2t^3$
  - $f(t) = t^2$
  - $f(t) = 1 - (1 - t)^2$
  - $f(t) = \sqrt{t}$
  - Больше примеров: easings.net

Анимации: сплайны

- Часто значения интерполируют, используя сплайны: кривые, позволяющие удобно настраивать зависимость некой величины от параметра `t`
- Обычно сплайн строится по набору точек и, возможно, значений производных в этих точках

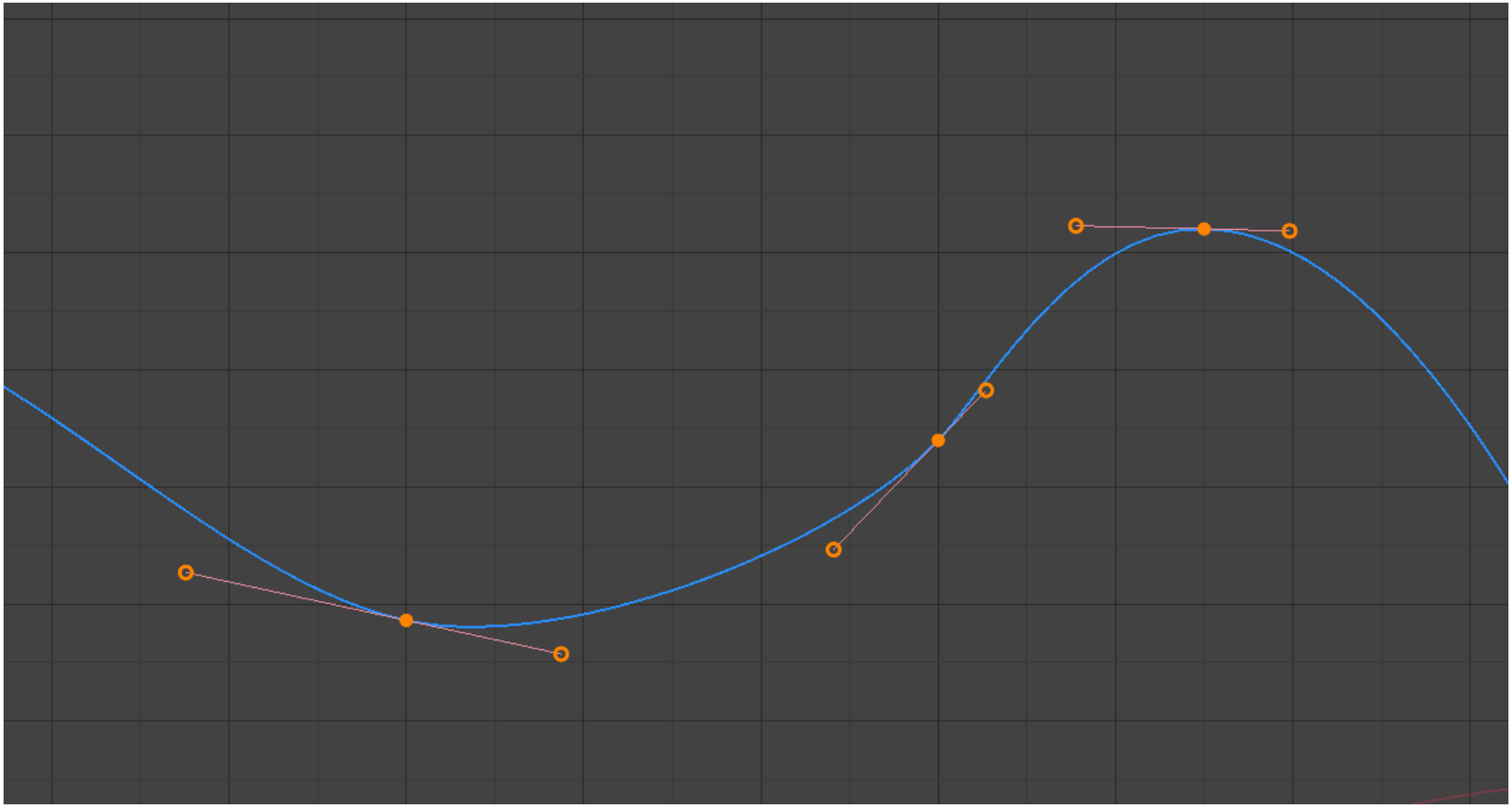
Анимации: сплайны

- Часто значения интерполируют, используя сплайны: кривые, позволяющие удобно настраивать зависимость некой величины от параметра `t`
- Обычно сплайн строится по набору точек и, возможно, значений производных в этих точках
- Виды сплайнов:
  - Сплайны Безье
  - Кубические сплайны
  - B-сплайны
  - NURBS
  - etc.

Анимации: сплайны

- Часто значения интерполируют, используя сплайны: кривые, позволяющие удобно настраивать зависимость некой величины от параметра `t`
- Обычно сплайн строится по набору точек и, возможно, значений производных в этих точках
- Виды сплайнов:
  - Сплайны Безье
  - Кубические сплайны
  - B-сплайны
  - NURBS
  - etc.
- N.B.: спектр применения сплайнов не ограничивается анимациями!
  - Curve fitting
  - Представление сложных геометрических форм (напр. зданий, шрифтов)
  - etc.

Анимации: сплайны



Bitmap-анимации

- Меняющееся со временем изображение

Bitmap-анимации

- Меняющееся со временем изоражение
- 3D текстура
  - Номер кадра - 3я текстурная координата (нормированная)
  - Интерполирует между кадрами

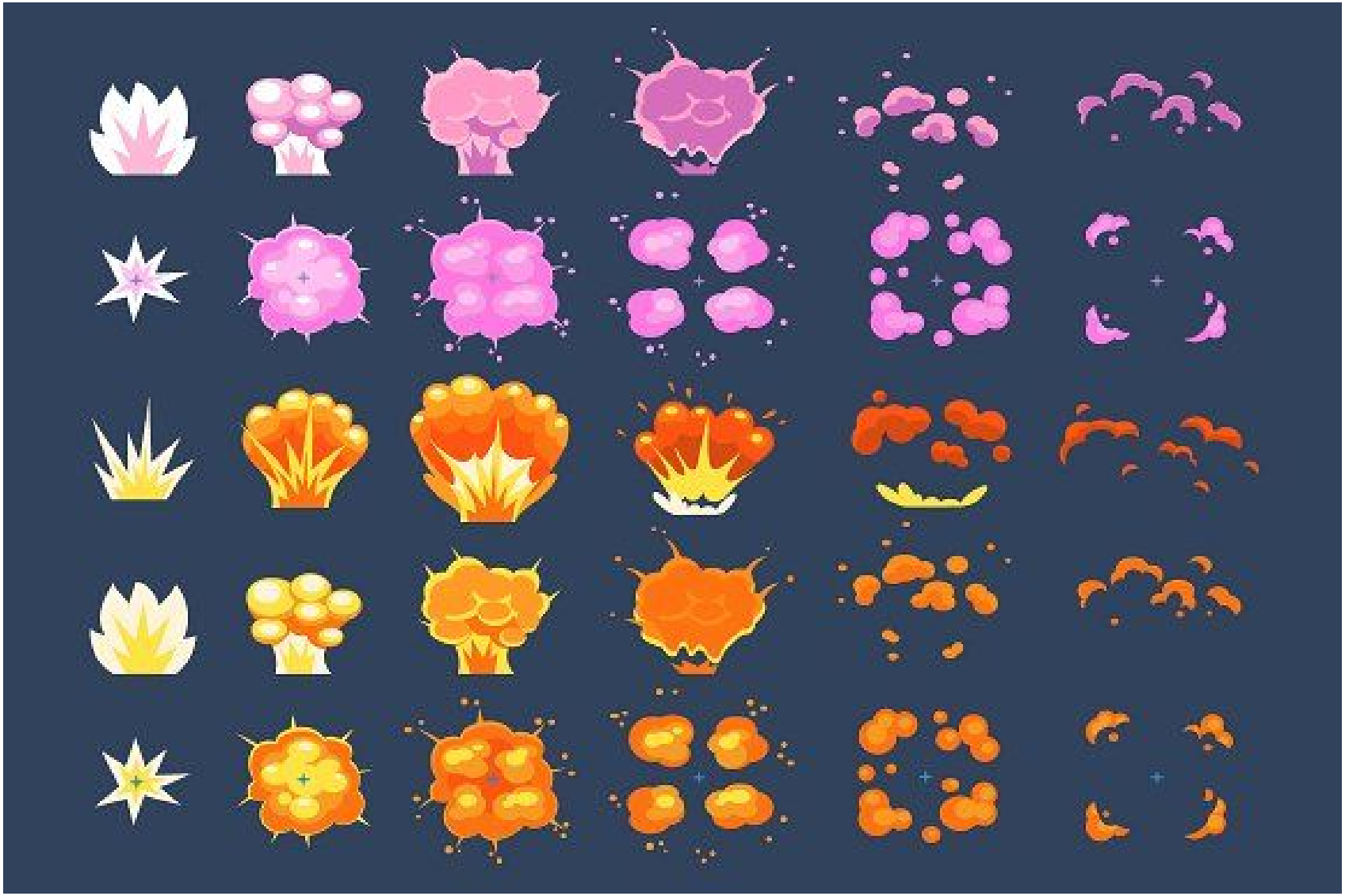
Bitmap-анимации

- Меняющееся со временем изоражение
- 3D текстура
  - Номер кадра - 3я текстурная координата (нормированная)
  - Интерполирует между кадрами
- 2D array текстура
  - Номер кадра - 3я текстурная координата (**не** нормированная)
  - **Не** интерполирует между кадрами

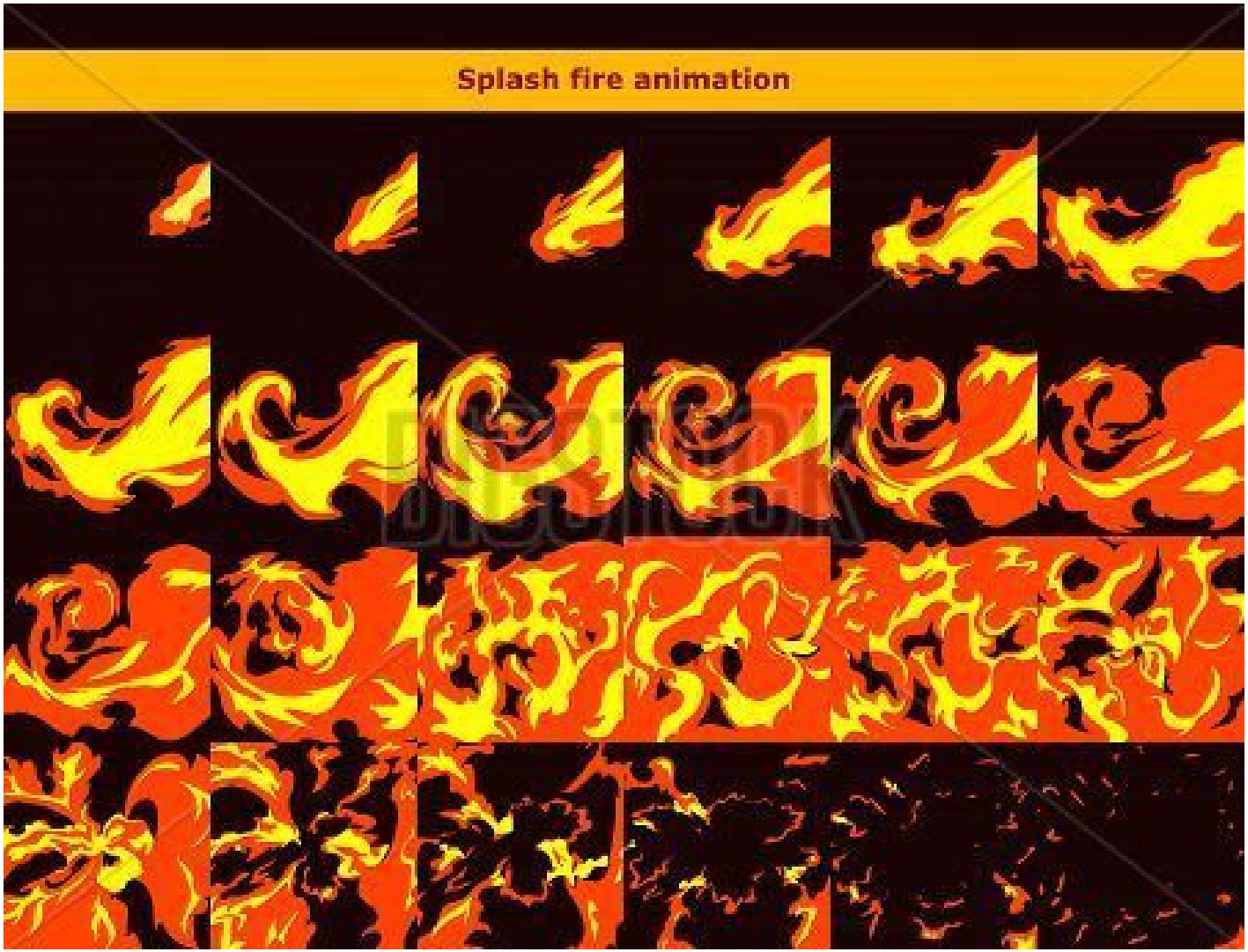
Bitmap-анимации

- Меняющееся со временем изоражение
- 3D текстура
  - Номер кадра - 3я текстурная координата (нормированная)
  - Интерполирует между кадрами
- 2D array текстура
  - Номер кадра - 3я текстурная координата (**не** нормированная)
  - **Не** интерполирует между кадрами
- 2D текстурный атлас - текстура, хранящая несколько изображений бок о бок
  - По номеру кадра вычисляются настоящие текстурные координаты
  - **Не** интерполирует между кадрами

Текстура-атлас с анимацией



Текстура-атлас с анимацией



Представление вращений

- Обычно мы представляли вращения матрицей
- Матрица 3x3 - 9 значений, это много
- Применить матрицу к вектору - минимум операций сложения и умножения
- Композиция вращений - произведение матриц, довольно быстрая операция

Представление вращений

- Обычно мы представляли вращения матрицей
- Матрица 3x3 - 9 значений, это много
- Применить матрицу к вектору - минимум операций сложения и умножения
- Композиция вращений - произведение матриц, довольно быстрая операция
- Вращения образуют 3х-мерную группу, т.е. описываются 3мя параметрами, например углами Эйлера
- Применить вращение, выраженное через углы Эйлера - много тригонометрических функций, медленно
- Композиция таких вращений - сложная операция, много тригонометрических функций

Представление вращений

- Обычно мы представляли вращения матрицей
- Матрица 3x3 - 9 значений, это много
- Применить матрицу к вектору - минимум операций сложения и умножения
- Композиция вращений - произведение матриц, довольно быстрая операция
- Вращения образуют 3х-мерную группу, т.е. описываются 3мя параметрами, например углами Эйлера
- Применить вращение, выраженное через углы Эйлера - много тригонометрических функций, медленно
- Композиция таких вращений - сложная операция, много тригонометрических функций
- Хочется компромисс между сложностью и объёмом хранения

Кватернионы

- Кватернионы $\mathbb{H}$ – четырёхмерная *некоммутативная* алгебра над вещественными числами с тремя мнимыми единицами
- Каждый элемент $q \in \mathbb{H}$ представляется в виде $q = a + bi + cj + dk$, где $a, b, c, d \in \mathbb{R}$ - коэффициенты кватерниона

Кватернионы

- Кватернионы $\mathbb{H}$ – четырёхмерная *некоммутативная* алгебра над вещественными числами с тремя мнимыми единицами
- Каждый элемент $q \in \mathbb{H}$ представляется в виде $q = a + bi + cj + dk$, где $a, b, c, d \in \mathbb{R}$ - коэффициенты кватерниона
- Правила умножения:
  - $i^2 = j^2 = k^2 = -1$
  - $ij = -ji = k$
  - $jk = -kj = i$
  - $ki = -ik = j$

Кватернионы

- Сопряжённый кватернион определяется как $\overline{q} = a - bi - cj - dk$
- Норма кватерниона: $q \cdot \overline{q} = a^2 + b^2 + c^2 + d^2 = |q|^2$
- Обратный кватернион: $q^{-1} = \frac{1}{|q|^2}\overline{q}$

Кватернионы: альтернативное представление

- Для кватерниона $q = a + bi + cj + dk$ назовём его скалярной частью число a, а векторной частью – вектор $v = (b, c, d)$
- Кватернион – пара скаляр + вектор: $q = (a, v)$

Кватернионы: альтернативное представление

- Для кватерниона $q = a + bi + cj + dk$ назовём его скалярной частью число a, а векторной частью – вектор $v = (b, c, d)$
- Кватернион – пара скаляр + вектор: $q = (a, v)$
- Произведение кватернионов:
  $(a_1, v_1) \cdot (a_2, v_2) = (a_1 \cdot a_2 - v_1 \cdot v_2, a_1 v_2 + a_2 v_1 + v_1 \times v_2$

Кватернионы: альтернативное представление

- Для кватерниона $q = a + bi + cj + dk$ назовём его скалярной частью число a, а векторной частью – вектор $v = (b, c, d)$
- Кватернион – пара скаляр + вектор: $q = (a, v)$
- Произведение кватернионов:
  $(a_1, v_1) \cdot (a_2, v_2) = (a_1 \cdot a_2 - v_1 \cdot v_2, a_1 v_2 + a_2 v_1 + v_1 \times v_2$
- В таком виде кватернионы удобно реализовывать в шейдерах

Кватернионы: вращения

- Представим произвольный трёхмерный вектор v как кватернион с нулевой скалярной частью $(0, v)$
- Вращение вокруг оси w (единичный вектор) на угол θ можно реализовать как $q \cdot (0, v) \cdot q^{-1}$, где $q = (\cos \frac{\theta}{2}, w \cdot \sin \frac{\theta}{2})$

Кватернионы: вращения

- Представим произвольный трёхмерный вектор v как кватернион с нулевой скалярной частью $(0, v)$
- Вращение вокруг оси w (единичный вектор) на угол θ можно реализовать как $q \cdot (0, v) \cdot q^{-1}$, где $q = (\cos \frac{\theta}{2}, w \cdot \sin \frac{\theta}{2})$
- N.B.: $q^{-1} = (\cos \frac{\theta}{2}, -w \cdot \sin \frac{\theta}{2})$

Кватернионы: вращения

- Представим произвольный трёхмерный вектор v как кватернион с нулевой скалярной частью $(0, v)$
- Вращение вокруг оси w (единичный вектор) на угол θ можно реализовать как $q \cdot (0, v) \cdot q^{-1}$, где $q = (\cos \frac{\theta}{2}, w \cdot \sin \frac{\theta}{2})$
- N.B.: $q^{-1} = (\cos \frac{\theta}{2}, -w \cdot \sin \frac{\theta}{2})$
- Только алгебраические операции $\Rightarrow$ быстро!

Кватернионы: вращения

- Представим произвольный трёхмерный вектор v как кватернион с нулевой скалярной частью $(0, v)$
- Вращение вокруг оси w (единичный вектор) на угол θ можно реализовать как $q \cdot (0, v) \cdot q^{-1}$, где $q = (\cos \frac{\theta}{2}, w \cdot \sin \frac{\theta}{2})$
- N.B.: $q^{-1} = (\cos \frac{\theta}{2}, -w \cdot \sin \frac{\theta}{2})$
- Только алгебраические операции $\Rightarrow$ быстро!
- Композиция вращений – произведение кватернионов: $q_2 \cdot (q_1 \cdot (0, v) \cdot q_1^{-1}) \cdot q_2^{-1} = (q_2 \cdot q_1) \cdot (0, v) \cdot (q_1^{-1} \cdot q_2^{-1}) = (q_2 q_1) \cdot (0, v) \cdot (q_2 q_1)^{-1}$

Кватернионы: вращения

- Представим произвольный трёхмерный вектор v как кватернион с нулевой скалярной частью $(0, v)$
- Вращение вокруг оси w (единичный вектор) на угол θ можно реализовать как $q \cdot (0, v) \cdot q^{-1}$, где $q = (\cos \frac{\theta}{2}, w \cdot \sin \frac{\theta}{2})$
- N.B.: $q^{-1} = (\cos \frac{\theta}{2}, -w \cdot \sin \frac{\theta}{2})$
- Только алгебраические операции $\Rightarrow$ быстро!
- Композиция вращений – произведение кватернионов: $q_2 \cdot (q_1 \cdot (0, v) \cdot q_1^{-1}) \cdot q_2^{-1} = (q_2 \cdot q_1) \cdot (0, v) \cdot (q_1^{-1} \cdot q_2^{-1}) = (q_2 q_1) \cdot (0, v) \cdot (q_2 q_1)^{-1}$
- Стандартный способ для представления вращений объектов в 3D движках

Кватернионы: ссылки

- en.wikipedia.org/wiki/Quaternion
- en.wikipedia.org/wiki/Quaternions_and_spatial_rotation
- en.wikipedia.org/wiki/Rotation_formalisms_in_three_dimensions

Преобразования объектов

- Часто для задания преобразования, применяемого к объекту, нам не нужна целиком матрица аффинного преобразования
- Обычно это поворот + масштабирование + сдвиг

Преобразования объектов

- Часто для задания преобразования, применяемого к объекту, нам не нужна целиком матрица аффинного преобразования
- Обычно это поворот + масштабирование + сдвиг
- Поворот – кватернион q
- Масштабирование – число s (изотропное масштабирование) или три числа (разный масштаб по разным осям)
- Сдвиг – вектор сдвига t

Преобразования объектов

- Часто для задания преобразования, применяемого к объекту, нам не нужна целиком матрица аффинного преобразования
- Обычно это поворот + масштабирование + сдвиг
- Поворот – кватернион q
- Масштабирование – число s (изотропное масштабирование) или три числа (разный масштаб по разным осям)
- Сдвиг – вектор сдвига t

$$v \mapsto s \cdot qvq^{-1} + t \tag{1}$$

Иерархия объектов

- Часто объекты сцены/мира образуют иерархическую структуру (шапка на человеке, человек в машине, машина на корабле)
- Удобно описывать преобразование (позиция + поворот + масштабирование) не относительно центра сцены/мира, а относительно родительского объекта
  - N.B.: обычно в такой ситуации есть один корневой объект - сцена

Иерархия объектов

- Часто объекты сцены/мира образуют иерархическую структуру (шапка на человеке, человек в машине, машина на корабле)
- Удобно описывать преобразование (позиция + поворот + масштабирование) не относительно центра сцены/мира, а относительно родительского объекта
  - N.B.: обычно в такой ситуации есть один корневой объект - сцена
- Нужно уметь вычислять итоговое преобразование объекта, т.е. композицию всех преобразований от корня до нашего объекта

Композиция преобразований объектов

$$(t_2, s_2, q_2) \cdot (t_1, s_1, q_1) \cdot v = s_2 q_2 (s_1 q_1 v q_1^{-1} + t_1) q_2^{-1} + t_2 =$$
$$= s_2 s_1 (q_2 q_1) v (q_2 q_1)^{-1} + s_2 q_2 t_1 q_2^{-1} + t_2 =$$
$$(s_2 q_2 t_1 q_2^{-1} + t_2, s_2 s_1, q_2 q_1) \cdot v \quad (2)$$

Анимация трёхмерных моделей

- Анимация положения объекта в пространстве – неплохо, но скучно
- Хочется анимировать сам объект, т.е. двигать его вершины

Анимация трёхмерных моделей

- Анимация положения объекта в пространстве – неплохо, но скучно
- Хочется анимировать сам объект, т.е. двигать его вершины
- 2 способа:
  - Покадровая анимация (keyframe animation, morph-target animation)
  - Скелетная анимация

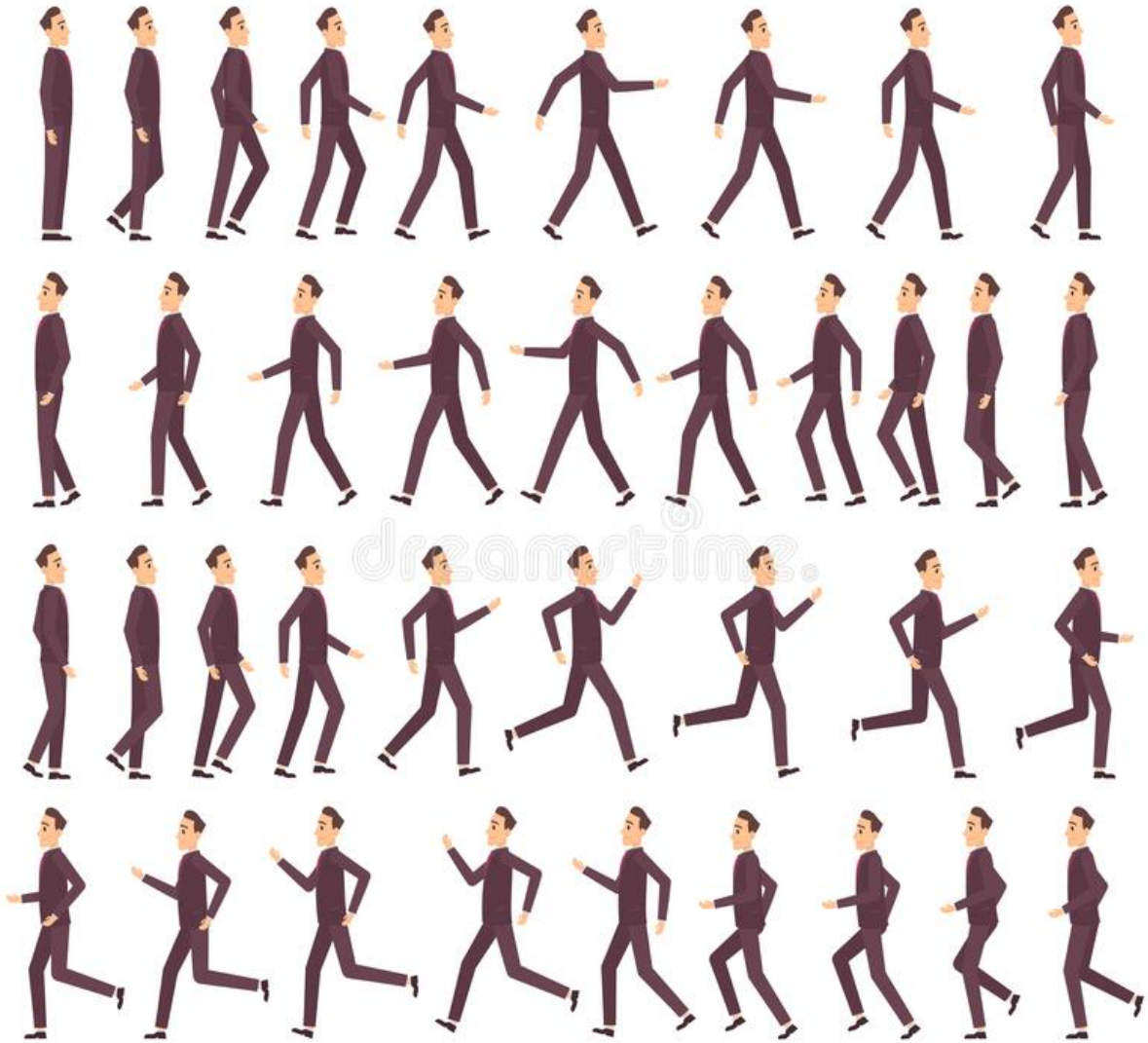
Покадровая анимация моделей

- Анимация хранится в виде "кадров": фиксированных состояний модели (наборов координат вершин)

Покадровая анимация моделей

- Анимация хранится в виде "кадров": фиксированных состояний модели (наборов координат вершин)
- Вершинный шейдер принимает два набора атрибутов позиций вершин и интерполирует между ними
  - N.B.: интерполяция может использовать easing

Покадровая анимация моделей



Покадровая анимация моделей

- Много вариантов реализации:
  - При смене кадра анимации загружать в VBO новые данные
  - При смене кадра менять VBO/VAO
  - Хранить кадры отдельно (например, в buffer textures), передавать в шейдер только номер кадра

Покадровая анимация моделей

- Много вариантов реализации:
  - При смене кадра анимации загружать в VBO новые данные
  - При смене кадра менять VBO/VAO
  - Хранить кадры отдельно (например, в buffer textures), передавать в шейдер только номер кадра
- Много проблем:
  - Сложно модифицировать: нужно двигать *все* вершины модели
  - Требует много памяти
  - Для хорошего качества нужно много кадров, иначе будут артефакты интерполяции (напр. модель начнёт пересекать саму себя)

Покадровая анимация моделей

- Много вариантов реализации:
  - При смене кадра анимации загружать в VBO новые данные
  - При смене кадра менять VBO/VAO
  - Хранить кадры отдельно (например, в buffer textures), передавать в шейдер только номер кадра
- Много проблем:
  - Сложно модифицировать: нужно двигать *все* вершины модели
  - Требует много памяти
  - Для хорошего качества нужно много кадров, иначе будут артефакты интерполяции (напр. модель начнёт пересекать саму себя)
- Обычно **не** используется для 3D моделей

Скелетная анимация моделей

- Модель привязывается к виртуальному "скелету"

Скелетная анимация моделей

- Модель привязывается к виртуальному "скелету"
- Скелет – иерархия виртуальных "костей"

Скелетная анимация моделей

- Модель привязывается к виртуальному "скелету"
- Скелет – иерархия виртуальных "костей"
- Каждая вершина привязана к одной или (чаще) нескольким костям

Скелетная анимация моделей

- Модель привязывается к виртуальному "скелету"
- Скелет – иерархия виртуальных "костей"
- Каждая вершина привязана к одной или (чаще) нескольким костям
- Каждой паре вершина-кость соответствует некоторый вес: насколько эта кость влияет на эту вершину (сумма весов для одной вершины должна равняться 1)

Скелетная анимация моделей

- Модель привязывается к виртуальному "скелету"
- Скелет – иерархия виртуальных "костей"
- Каждая вершина привязана к одной или (чаще) нескольким костям
- Каждой паре вершина-кость соответствует некоторый вес: насколько эта кость влияет на эту вершину (сумма весов для одной вершины должна равняться 1)
- Кадры анимации задаются только для скелета

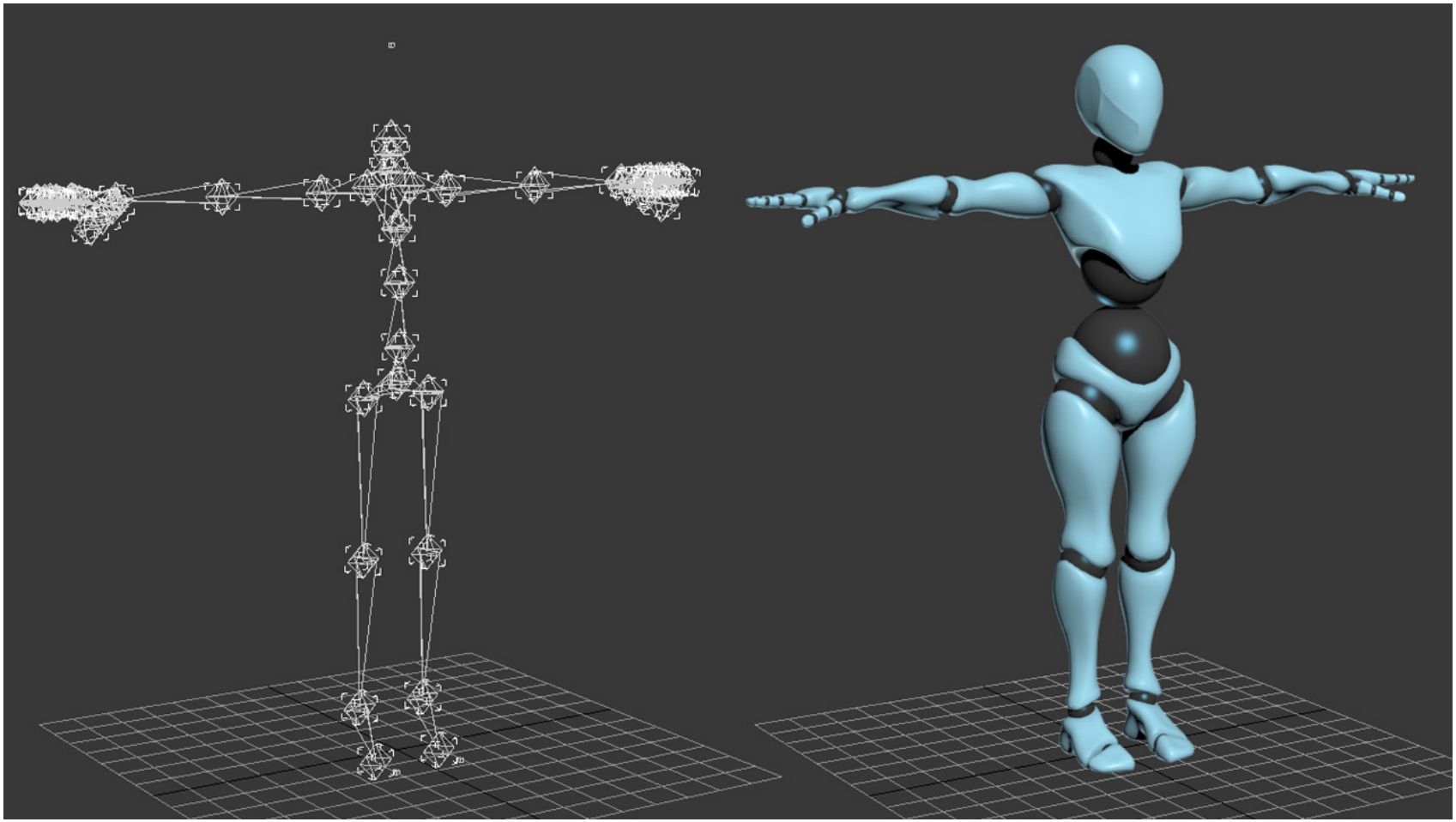
Скелетная анимация моделей

- Модель привязывается к виртуальному "скелету"
- Скелет – иерархия виртуальных "костей"
- Каждая вершина привязана к одной или (чаще) нескольким костям
- Каждой паре вершина-кость соответствует некоторый вес: насколько эта кость влияет на эту вершину (сумма весов для одной вершины должна равняться 1)
- Кадры анимации задаются только для скелета
- Интерполируются только преобразования костей (костей гораздо меньше, чем вершин $\Rightarrow$ это не страшно делать даже на CPU)

Скелетная анимация моделей

- Модель привязывается к виртуальному "скелету"
- Скелет – иерархия виртуальных "костей"
- Каждая вершина привязана к одной или (чаще) нескольким костям
- Каждой паре вершина-кость соответствует некоторый вес: насколько эта кость влияет на эту вершину (сумма весов для одной вершины должна равняться 1)
- Кадры анимации задаются только для скелета
- Интерполируются только преобразования костей (костей гораздо меньше, чем вершин $\Rightarrow$ это не страшно делать даже на CPU)
- В вершинном шейдере вычисляется итоговое преобразование для вершины как среднее между преобразованиями связанных с ней костей

Скелетная анимация моделей



Скелетная анимация моделей

```
uniform mat4x4 bones[16];

layout (location = 0) in vec4 position;
layout (location = 1) in ivec2 bone_ids;
layout (location = 2) in vec2 bone_weights;

void main()
{
  gl_Position =
      bone_weights.x * bones[bone_ids.x] * position
    + bone_weights.y * bones[bone_ids.y] * position;
}
```

Скелетная анимация моделей

- К нормалям тоже нужно применять преобразования (но не сдвиги!)

Скелетная анимация моделей

- К нормалям тоже нужно применять преобразования (но не сдвиги!)
- Кости обычно тоже выстроены в иерархию $\Rightarrow$ перед применением нужно вычислить суммарное преобразование (композицию)
  - Нужно быть внимательным к особенностям задания преобразований в разных редакторах и форматах

Скелетная анимация моделей

- Удобно и интуитивно модифицировать (все 3D-редакторы имеют поддержку скелетных анимаций)

Скелетная анимация моделей

- Удобно и интуитивно модифицировать (все 3D-редакторы имеют поддержку скелетных анимаций)
- Небольшой расход памяти (модель не дублируется)

Скелетная анимация моделей

- Удобно и интуитивно модифицировать (все 3D-редакторы имеют поддержку скелетных анимаций)
- Небольшой расход памяти (модель не дублируется)
- Самый распространённый способ анимировать модели

Скелетная анимация

- Преобразования для скелета часто комбинируют с процедурными элементами, которые невозможно заранее сделать в 3D-редакторе

Скелетная анимация

- Преобразования для скелета часто комбинируют с процедурными элементами, которые невозможно заранее сделать в 3D-редакторе
  - Голова человека поворачивается в сторону собеседника
  - Нога человека встаёт на камень
  - Рука человека берёт предмет

Скелетная анимация

- Преобразования для скелета часто комбинируют с процедурными элементами, которые невозможно заранее сделать в 3D-редакторе
  - Голова человека поворачивается в сторону собеседника
  - Нога человека встаёт на камень
  - Рука человека берёт предмет
- Для этого нужно решать обратную задачу: хотим, чтобы кость была в известной точке, нужно найти преобразования, которыми этого можно добиться
  - Inverse kinematics

Скелетная анимация

- Есть способы для некоторых ситуаций полностью процедурно генерировать преобразования скелета

Скелетная анимация

- Есть способы для некоторых ситуаций полностью процедурно генерировать преобразования скелета
- Анимация передвижения пауков: youtube.com/watch?v=e6Gjhr1IP6w

Скелетная анимация

- Есть способы для некоторых ситуаций полностью процедурно генерировать преобразования скелета
- Анимация передвижения пауков: youtube.com/watch?v=e6Gjhr1IP6w
- Оффлайн генерация анимации движения для двуногих: youtube.com/watch?v=pgaEE27nsQw

Скелетная анимация: ссылки

- en.wikipedia.org/wiki/Skeletal_animation
- learnopengl.com/Guest-Articles/2020/Skeletal-Animation
- ogldev.org/www/tutorial38/tutorial38.html
- youtube.com/watch?v=f3Cr8Yx3GGA