



НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ

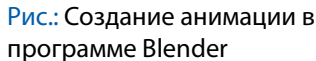
Факультет компьютерных наук
Департамент программной инженерии
Курсовая работа

Программа скелетная анимация

Выполнил студент группы 151БПИ
Абрамов Артем Михайлович
Научный руководитель:
доцент департамента программной
инженерии, к.т.н
Ахметсафина Римма Закиевна

2016

Отображение анимации - одна из наиболее актуальных задач в производственной, научной и деловой сферах, а также в области развлечений.





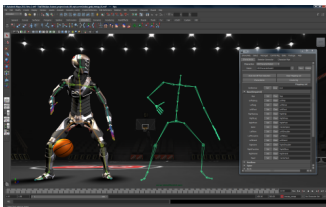
???

Цель работы - реализовать систему неявной скелетной анимации.

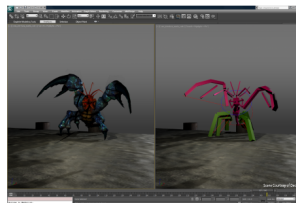
Задачи работы

1. Создание исходных данных (файлов) для неявной скелетной анимации.
2. Загрузка анимации из файла (содержание описанно в ТЗ).
3. Расчет кадров анимации.
4. Воспроизведение анимации на экране средствами OpenGL.

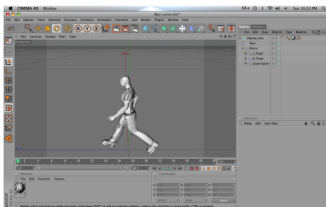
Пакеты 3-х мерного моделирования. Maya, 3ds MAX, Blender, Cinema4D.



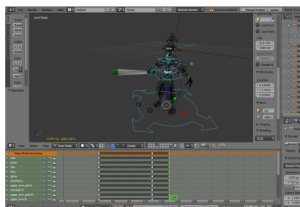
Maya



3ds MAX



Cinema4D

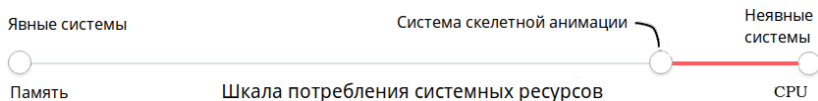


Blender

Для 3-х мерной анимации не существует оптимального подхода. Все системы балансируют между методами с большим количеством вычислений и методами, требующими большого объема памяти.

Неявные системы используются, когда действия персонажа связаны с другими предметами, и нельзя предугадать все возможные варианты анимации. Например для того, чтобы ставить ступню параллельно поверхности при движении по неровной земле.

Предпочтение **явным системам** отдается, когда необходимо анимировать большие группы людей или животных.



Явная система - хранение отдельной модели для каждого кадра.
После записи данных в файл существует много методов для воспроизведения анимации. Такие методы требуют лишь элементарной математики.
Однако типичная запись одного трэка анимации для одного персонажа занимает около 10MB (в формате MD3).



Рис.: Каждому кадру соответствует своя модель

Неявная система - хранение не моделей, а более высокоуровневого описания движения.

В частности **неявные системы скелетной анимации** содержат описание (через матрицу поворота) для каждой кости, как например локоть, плечо, шея. В реальном времени эти описания применяются к неанимированной модели для расчета следующего кадра анимации. Эти расчеты обычно требуют сложной математики с матрицами и тригонометрией. А следовательно и много CPU времени.

Для реализации требуются 3 вещи: скелет, модель и трэк анимации.

Скелет определяет иерархию частей тела персонажа, в трэке содержатся матрицы для поворота скелета в ключевые моменты времени.

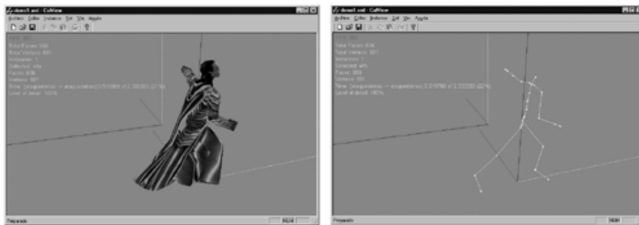


Рис.: Слева: анимированный персонаж; справа: скелет для данного кадра

Матрицы поворота для всех костей записываются в трэке относительно матрицы поворота родителя. Поэтому для деформации скелета необходимо применять матрицы последовательно. Мы начинаем с корневой кости и применяем к ней описанную в трэке анимации матрицу поворота. Затем, мы двигаемся вглубь скелета применяя матрицу родителя и деформацию описанную в трэке (то есть считаем глобальную матрицу поворота для данной кости).

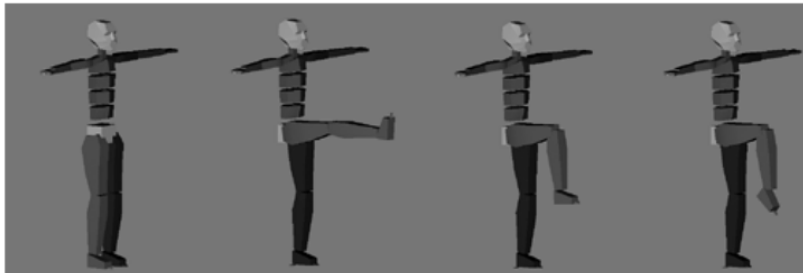


Рис.: Применение преобразований, начиная от копчика (корневой кости) и заканчивая ступней.

После того, как рассчитаны матрицы поворотов для скелета, их необходимо применить на вершины модели. Для этого используется рекурсивный алгоритм очень похожий на предыдущий.

```
deform (bone root , mesh original , mesh deformed)
  for each child_bone of root
    for each vertex in the original mesh
      if bone_weight > 0
        apply bone global transform to vertex
        scale the resulting point by the bone weight
        store the result in processeddata
      end if
    end for
    if child_bone has children
      deform (children of this node , mesh original , deformed)
    end if
  end for
```

- Язык программирования C#
- Библиотека Assimp v3.1 (<http://assimp.org/>) для чтения файлов в формате collada (.dae).
- Библиотека OpenTK v1.1.4 (<http://www.opentk.com/>) для вызова функций OpenGL из C# и предоставления базовых классов, например: Matrix4, Vector3.

Диаграмма основных блоков.

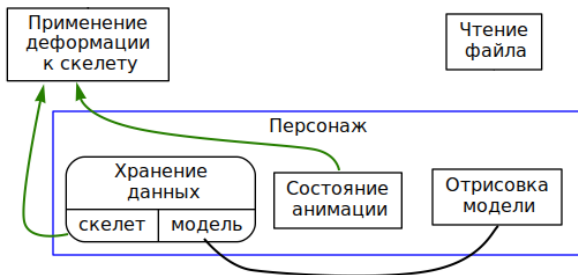


Рис.: Схема логических блоков в программе

Заполнение структур данными из файла.

С помощью библиотеки Assimp производим чтение из файла. Для оптимальной работы мы переводим данные в свои структуры. Другие функции этой библиотеки не используются.

```
public void LoadScene(byte[] filedata)
{
    using (MemoryStream fs = new MemoryStream(filedata))
    {
        _cur_scene = new SceneWrapper(ReadAssimpScene(fs, "dae"));
        _action = new Animator(_cur_scene.Animations[0]);
        BoneNode bones = _cur_scene.BuildBoneNodes("Armature");
        Node mesh = _cur_scene.FindNode("Mesh");
        ActionState state = new ActionState();
        _enttity = new Entity(_cur_scene, mesh, bones, state);
    }
}
```

Хранит состояние анимации. Наиболее важные поля:

- Название трэка анимации.
- Настоящий момент времени в секундах.
- Индексы всех ключевых кадров и время каждого ключевого кадра.

Есть функция `SetTime(...)` для перехода к определенному моменту времени. Она находит интервал между ключевыми кадрами, подсчитывает величину интерполяции.

Работает со скелетом и моделью. Реализует функции поиска костей в скелете или подмоделей в модели.

Функция BuildBones строит скелет по данным из модели (скелет как отдельный класс не существует, он определяется корневой костью).

```
class BoneNode
{
    public string Name;
    public Matrix4 GlobalTransform;
    public Matrix4 LocalTransform;

    public BoneNode Parent;
    public List<BoneNode> Children;
    public BoneNode(Node assimp_node) { ... }
}
```

Рис.: Класс описывающий кость скелета

Применяет данные, описывающие (в матрицах поворота) новую позицию для каждой кости к костям из скелета. То есть, деформирует скелет в соответствии с моментом времени в анимации.

На вход блока подается класс `ActionState`, содержащий информацию о состоянии анимации и корневая кость скелета.

Загружает данные о модели в OpenGL.

Запрашивает OpenGL об отводе буферов памяти под вершины, нормали, цвета вершин и массив индексов.

Применяет свойства материала, например: цвет, коэффициент рассеивания света, коэффициент свечения и т.д.

Позволяет запросить у OpenGL указатель на созданные буфера памяти для их модификации.

Объединяет компоненты необходимые для анимации одного персонажа. Хранит ссылки на скелет (корневую кость), состояние анимации (ActionState), на саму модель и на класс отрисовки модели (MeshDraw)

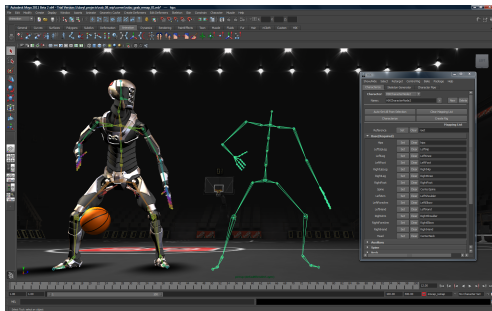
В частности блок персонажа применяет трансформации из скелета к вершинам модели (взвешивая действие каждой кости на вершину) и модифицирует данные в буфере данных OpenGL, что и создает эффект анимации.

Демонстрация

???

Пути дальнейшей работы:

1. Загрузка нескольких моделей
2. Наложение матрицы трансформации на отдельные модели
3. Выбор из нескольких трэков анимации
4. Нанесение текстур
5. Анимация на GPU



- Порев В.Н. Компьютерная графика. – СПб.: БХВ-Петербург, 2002. – 432 с.: ил.
- Daniel S.D. Core Techniques and Algorithms in Game Programming/S.D. Daniel. - Springer, 2008 - 727 с.
- Документация OpenGL 3.3 [Электронный ресурс] // <https://www.opengl.org/sdk/docs/man/> (Дата обращения: 21.10.2015, режим доступа: свободный)
- Рождерс Д. Алгоритмические основы машинной графики: Пер. с англ. - М.: Мир, 1989 - 512 с.

Факультет компьютерных наук
Департамент программной инженерии
Курсовая работа

Выполнил студент группы 151БПИ
Абрамов Артем Михайлович
Научный руководитель:
доцент департамента программной
инженерии, к.т.н
Ахметсафина Римма Закиевна

2016