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

Предметная область

3-х мерная компьютерная анимация - вид мультипликации, создаваемый при помощи компьютера.

В отличии от 2-х мерной анимаци, художник не рисует каждый кадр, а работает с моделью для которой последовательно задает различные позы.

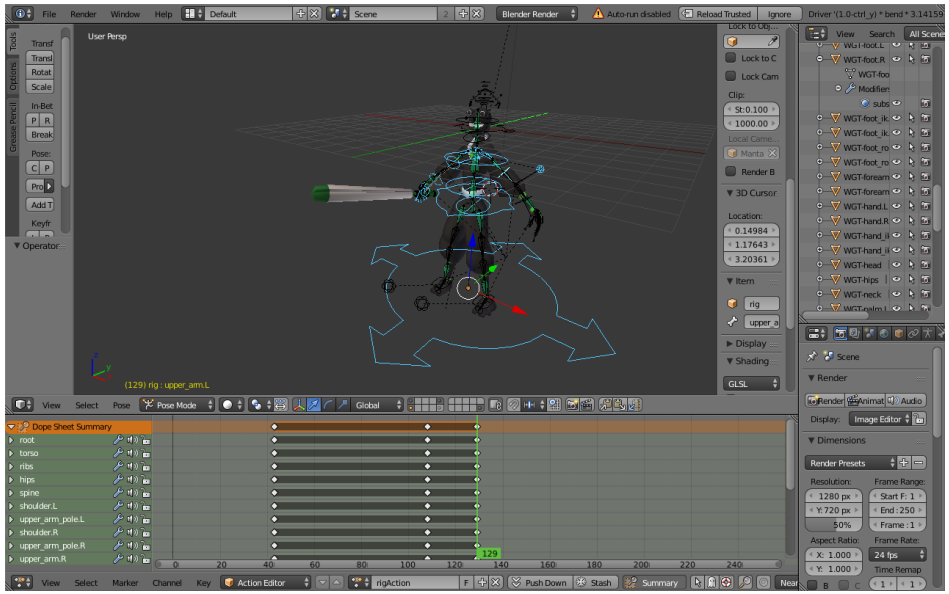
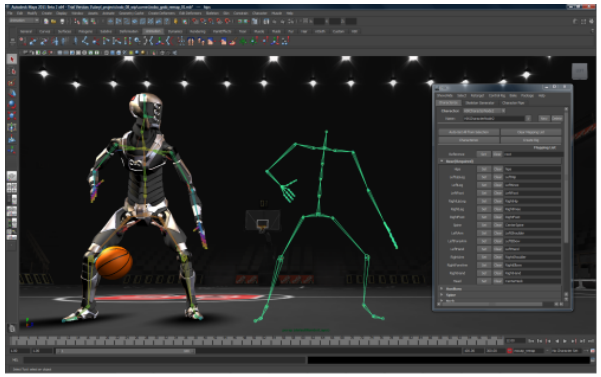


Рис.: Создание анимации в программе Blender

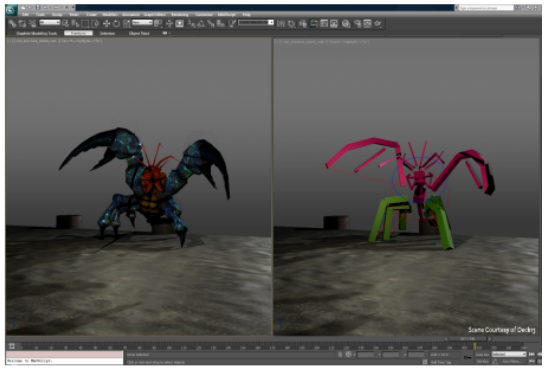
Основные определения

Обоснование актуальности работы

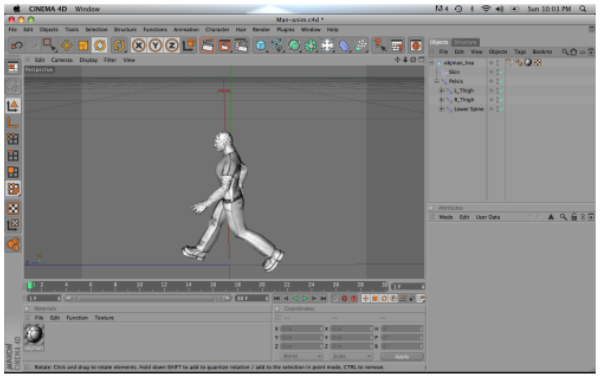
Отображение анимации - одна из наиболее актуальных задач в производственной, научной и деловой сферах, а также в области развлечений.



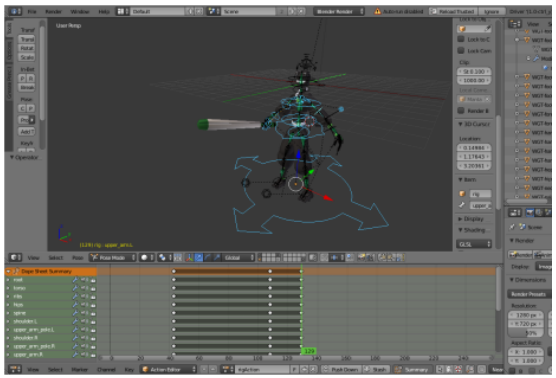
Maya



3ds MAX



Cinema4D



Blender

Цели и задачи работы

Цель работы - реализовать программу скелетной анимации.

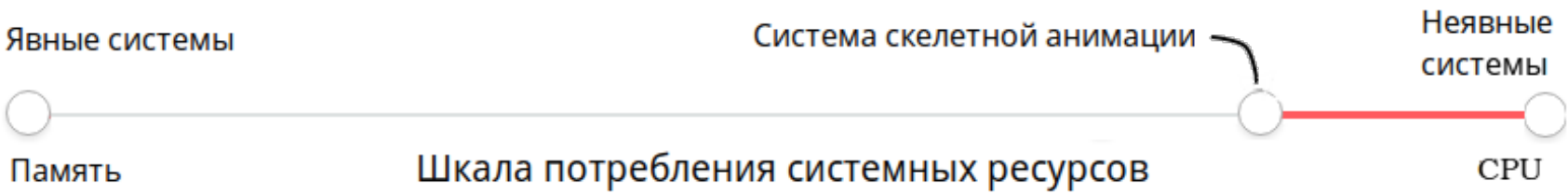
Задачи работы

1. Рассмотреть различные подходы к анимации 3D моделей.
2. Разработать структуры данных для хранения анимационных данных.
3. Выбрать технологии для реализации.
4. Понять алгоритм скелетной анимации.
5. Изучить технологию OpenGL.
6. Реализовать программу.

Различные подходы

Неявные системы используются, когда персонаж может совершать несколько действий одновременно и невозможно предугадать все возможные варианты анимации.

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



Явные системы анимации

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

Неявные системы анимации

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 рассчета следующего кадра анимации. Эти рассчеты обычно требуют сложной математики с матрицами и тригонометрией. А следовательно и много CPU времени.

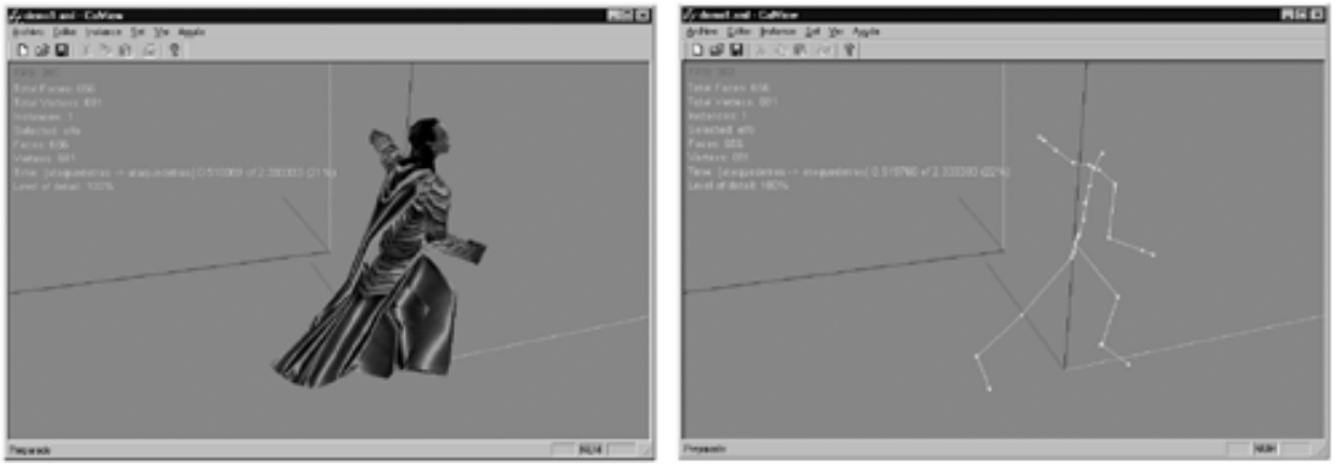


Рис.: Слева: анимированный персонаж; справа: скелет для данного кадра

Структуры данных: Кость и Скелет

Кость содержит информацию о трех мерной трансформации (которая состоит из поворота, растяжения и смещения), а также информацию о кости-отце. Глобальная матрица для кости-потомка, - это произведение глобальной матрицы кости-родителя и матрицы кости-потомка.

Скелетом называют иерархичную структуру сформированную костями. Скелет определяеться с помощью корневой кости в иерархии.

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

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

Структуры данных: Трек анимации

В треке содержатся матрицы поворота скелета в ключевые моменты времени. В упрощенном виде трек можно представить в качестве массива пар:
{ время ключевого кадра, массив из матриц поворота для костей }

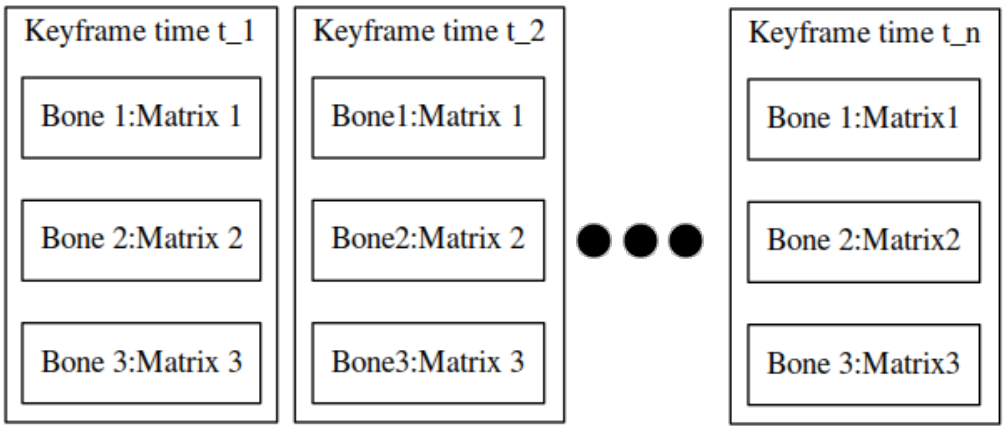


Рис.: В ключевые моменты времени (t_1, t_2, ... t_n), каждая кость ставиться в соответствие с матрицей локальной трансформации.

Структуры данных: Модель

Модель состоит из набора вершин и весов вершин. В пакете для трех мерной анимации каждая вершина модели «привязывается» к какой-либо кости скелета. Движение кости должно влиять на привязанные к ней вершины.

```
struct Vbo
{
    public int QuantityIds;
    public int VertexBufferId;
    public int NormalBufferId;
    public int ElementBufferId;
}
```

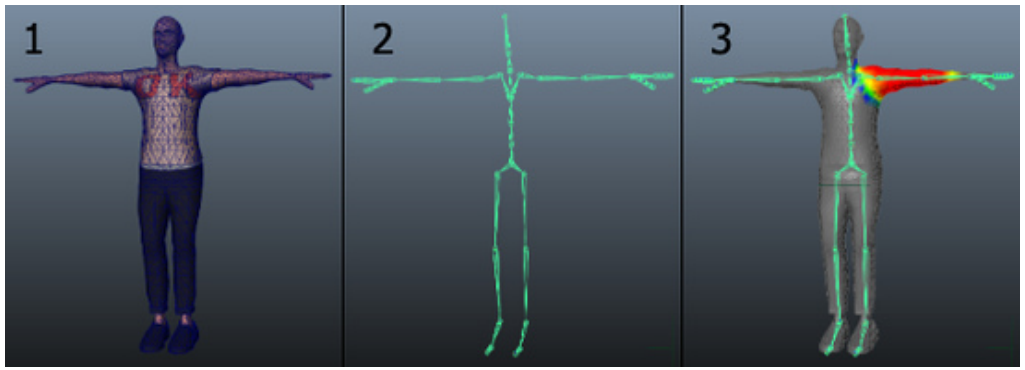


Рис.: №1 - модель; №2 - скелет; №3 - вершины модели, которые были привязанны к кости правого предплечья, выделенны красным

Алгоритм

В зависимости от времени из трека анимации извекаються данные о ключевом кадре. Связь матриц с костями известна и можно применить извлеченные матрицы к скелету. Далее, так как известна связь костей с вершинами, можно применить трансформации костей к вершинам модели.

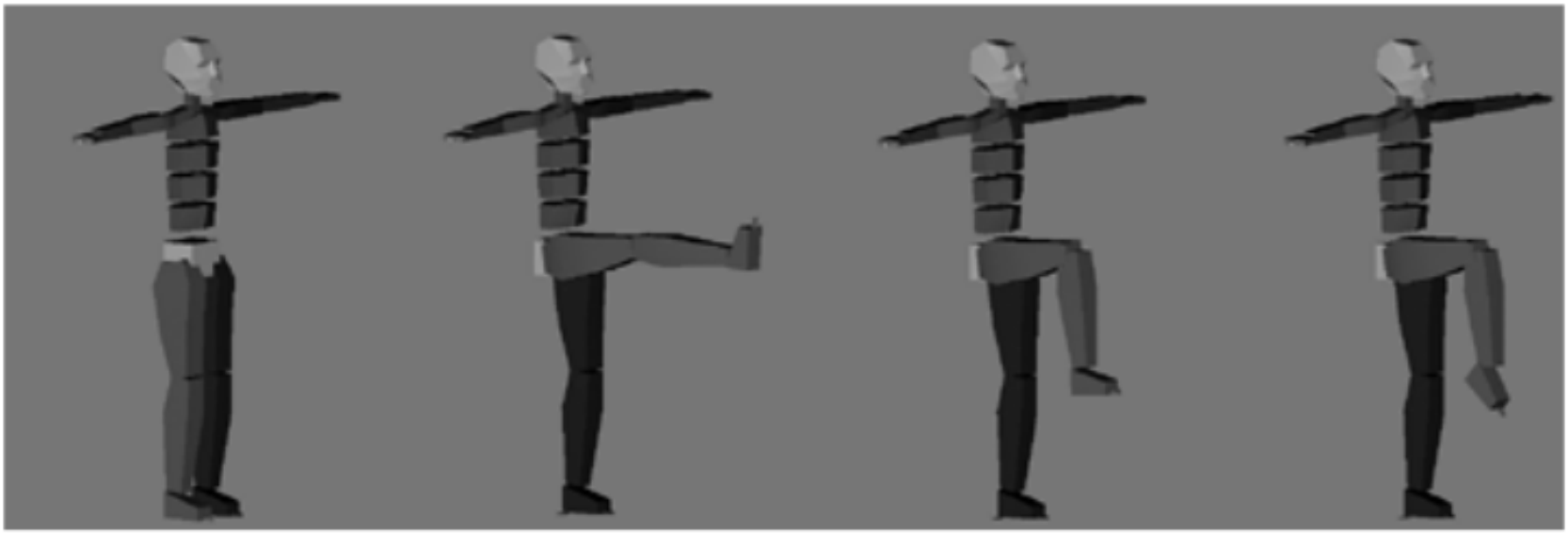


Рис.: Применение преобразований, начиная от копчика (корневой кости) и заканчивая ступней.

Применение трансформаций к скелету

В треке матрицы поворота записаны относительно матрицы поворота родителя. Поэтому для анимации необходимо применять матрицы последовательно. Начиная с корневой кости, примнить к ней описанную в треке анимации матрицу поворота. Затем, двигаться вглубь скелета по иерархии и находить произведение матриц родителя и потомка (извлеченной из трека). В псевдокоде:

```
deform (bone, global matrix, track matrices)
  get matrix for bone from track matrices
  update global matrix
  store the result in bone as global transform
  if root has children
    deform (children of this node, global, track matrices)
  end if
end function
```

Применение трансформаций к модели

После того, как рассчитанны матрицы поворотов для скелета, их необходимо применить на вершины модели.

```
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
```

Технологии и инструменты реализации

- 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 структур, например: Matrix4, Vector3.

Буферы памяти OpenGL

Загрузка данных о модели в OpenGL. Запрос OpenGL об отводе буферов памяти под вершины, нормали и массив индексов.

```
private void NewOpenGLBuf(out int bufId, List<Vector3D> data)
{
    GL.GenBuffers(1, out bufId);
    GL.BindBuffer(BufferTarget.ArrayBuffer, bufId);
    var byteCount = dataBuffer.Count * 12;
    var temp = new float[byteCount];
    foreach(var v in dataBuffer)
    {
        WriteIntoArray(temp, v);
    }
    GL.BufferData(BufferTarget.ArrayBuffer, (IntPtr)byteCount
            , temp, BufferUsageHint.StreamDraw);
}

private void Upload(out Vbo vboToFill)
{
    vboToFill = new Vbo();
    NewOpenGLBuf(out vboToFill.VertexBufferId, _mesh.Vertices);
}
```

Материалы в OpenGL

Загрузка данных о модели в OpenGL. Применение свойства материала, например: цвет, коэффициент рассеивания света, коэффициент свечения и т.д.

```
private void UseMaterial()
{
  color = Color2OpenTK(_material.ColorDiffuse);
  GL.Material(MaterialFace.FrontAndBack, Material.Diffuse, color);
  color = Color2OpenTK(_material.ColorAmbient);
  GL.Material(MaterialFace.FrontAndBack, Material.Ambient, color);

  color = new Color4(0, 0, 0, 1.0f);
  if (_material.HasColorEmissive)
  {
      color = color2OpenTK(_material.ColorEmissive);
  }
  GL.Material(MaterialFace.FrontAndBack, Material.Emission, color);
}
```

Результаты работы

Демонстрация

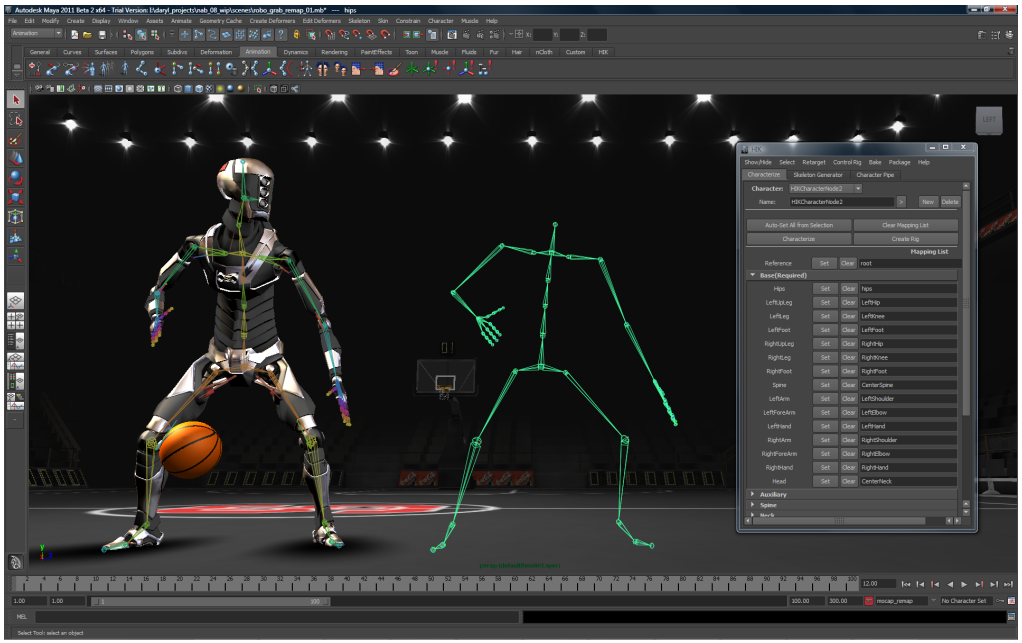
Апробация работы

???

Выводы по работе

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



Список использованных источников

- 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 анг. - М.: Мир, 1989 - 512 с.

Спасибо за внимание!

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