Реализация проекции сферического зеркала

Виноградов Владислав Игоревич, ННГУ им. Лобачевского, факультет ВМК. vlad.vinogradov47@gmail.com

Любов Дмитрий Викторович, ННГУ им. Лобачевского, факультет ВМК. dmitriy.lyubov@gmail.com

Носов Сергей Николаевич, ННГУ им. Лобачевского, факультет ВМК. sergei.nosov@gmail.com

Майоров Алексей Юрьевич, ННГУ им. Лобачевского, факультет ВМК. mayorov.alexey@gmail.com

АННОТАЦИЯ

В данной статье рассматривается вопрос адаптации медиа-контента, созданного для цифровых планетариев, под полнокупольные системы, использующие сферическое зеркало. Описывается программный комплекс, предоставляющий возможность калибровки проекционной аппаратуры и воспроизведения видео на сферическом экране.

Ключевые слова: цифровые планетарии, полнокупольные системы, сферический экран, fisheye-проекция, сферическое зеркало.

1 ВВЕДЕНИЕ

В настоящее время все большую популярность набирают цифровые полнокупольные системы. Это системы, в которых изображение проецируется на экран в форме полусферы. Изначально в качестве подобных экранов использовались купола планетариев. Сегодня широко доступны портативные полнокупольные системы, которые могут быть установлены в небольших помещениях. Такие системы чаще всего представляют собой надувной купол небольшого диаметра (от 3 до 8 метров), один проектор и специальное сферическое зеркало

Рис. 1 Проектор и сферическое зеркало

Так же на данный момент создано достаточное количество полнокупольного медиа-контента, который в основном состоит из научных и научно-популярных фильмов и видео-роликов. Так как все эти материалы предназначены для показа на сферическом экране, то при их съемке используется специальная fisheye - камера, обеспечивающая обзор 360°x180°. Получившиеся при этом кадры имеют fisheye - проекцию (Рис. 2). Данные кадры можно напрямую без конвертирования использовать при показе на сферических экранах с многопроекторными или fisheye-проекционными системами. Однако они не подходят для систем, использующих сферическое зеркало из-за отличий в проекции. Ввиду этого остро стоит задача адаптации имеющегося контента в fisheye-проекции к показу на малом куполе с использованием сферического зеркала (Рис. 2).

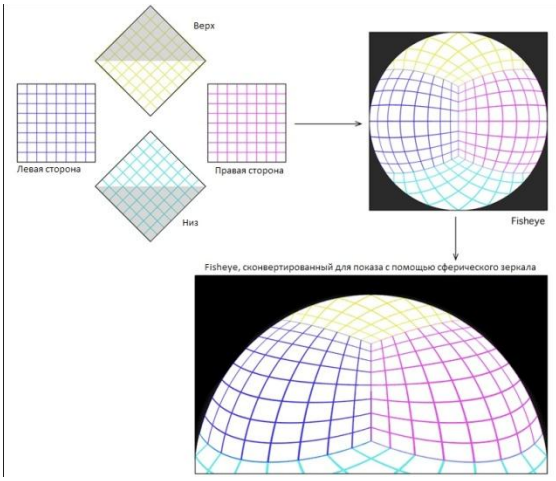

Рис. 2 Fisheye - проекция и конвертированный для сферического зеркала вариант

Также важной задачей является обеспечение возможности показа на сферическом экране стандартного видео, предназначенного для плоского экрана.

2 ПОСТАНОВКА ЗАДАЧИ

Конечной целью является создание программного комплекса, позволяющего решать следующие задачи:

1. калибровка проекционной аппаратуры полнокупольной системы (купол - сферическое зеркало - проектор);
2. воспроизведение видео, изначально находящегося в fisheye-формате.
3. Воспроизведение видео, предназначенного для показа на плоских экранах.

При создании программы калибровки необходимо выделить множество параметров, однозначно описывающих конфигурацию аппаратуры. Также необходимо определиться

с форматом, в котором будут храниться и использоваться результаты работы программы.

Для решения второй задачи – воспроизведения видео на куполе – было решено дополнить необходимой функциональностью один из существующих видеоплееров. В качестве такого плеера был выбран медиаплеер VLC, имеющий открытый исходный код. Его преимущество заключается в возможности расширения функционала за счет плагинов. Следовательно, необходимо реализовать в виде плагина видеофильтр для преобразования fisheye-видео. Данный плагин должен загружать информацию, полученную от программы калибровки, и осуществлять необходимое преобразование исходного изображения.

Для решения последней задачи медиа-контент, предназначенный для плоского экрана, необходимо перевести в проекцию fisheye. Следовательно необходимо разработать алгоритм, позволяющий производить подобное преобразование без потери качества и искажения исходного изображения.

3 РЕАЛИЗАЦИЯ ПРЕОБРАЗОВАНИЯ ИЗОБРАЖЕНИЯ

3.1 Моделирование полнокупольной системы

Первым делом обозначим множество параметров, которые однозначно определяют состояние конкретной полнокупольной системы. Выделим подобные множества для каждого из элементов системы. Начнем с купола:

- Dr Радиус купола. Является главной характеристикой купола.
- Dp = (0, 0, 0) - Центр купола. Для удобства расположим начало координат в центре купола.
- Da Угол наклона купола. Купол может располагаться не только параллельно полу, но и под наклоном вплоть до 90 градусов (вертикальный купол).

Далее рассмотрим параметры второй существенной части системы - сферического зеркала:

- Mr Радиус зеркала.
- Mp Положение центра зеркала в системе координат купола.

Теперь рассмотрим параметры проектора:

- Pp Положение проектора в системе координат купола.
- euler вектор углов Эйлера, отражающих возможные наклоны проектора.
- ThrowRatio Отношение расстояние от проектора до экрана к ширине получаемого изображения.

И наконец рассмотрим параметры входного и выходного изображений:

- angle Угол поворота исходного изображения вокруг своего центра.
- width Требуемая ширина выходного изображения.
- height Требуемая высота выходного изображения.
- SF Скалирующий множитель.
- mirrored Зеркальность относительно вертикальной оси.

Упрощенная схема полнокупольной системы представлена на рис. 1.

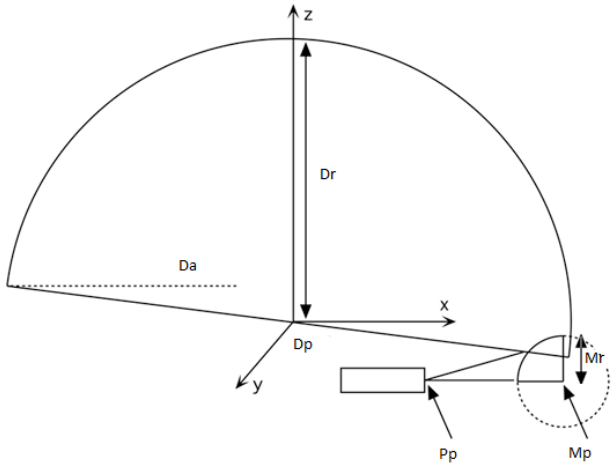

Рис. 3 Схема полнокупольной системы

3.2 Алгоритм преобразования fishye-кадра к кадру, использующему проекцию сферического зеркала

После того, как мы определили все необходимые для вычислений параметры, перейдем к описанию алгоритма преобразования. За основу возьмем известный метод трассировки лучей и проследим путь луча от проектора до поверхности купола. Функция генерации луча, отвечающего за проецируемый пиксель (i, j) может быть описана следующим псевдо-кодом:

```
GenerateRay (Ray ray, Vec3 view, Vec3 up,
             Vec3 side, i, j)
{
  Real aspect = width/height;
  Vec2 scale = {1/ThrowRatio, aspect/ThrowRatio}
  Vec2 screen = { i/height * scale.x;
                  (j/width - 0.5) * scale.y }
  ray.Direction = view + side * screen.x +
                  up * screen.y
  ray.Origin = Pp
}
```

Листинг 1 Генерация луча

Здесь вектора view, up, side задают положение проектора и вычисляются с использованием углов Эйлера проектора.

Следующим шагом после генерации луча является его трассировка. Для каждого луча нужно найти точку его пересечения с зеркалом, а затем вычислить отраженный луч и найти точку его пересечения с куполом - Q. Точка пересечения луча и сферы может быть найдена из решения квадратного уравнения.

Далее необходимо по координатам точки на куполе восстановить ее координаты на исходном fisheye-

изображении. Ниже приведен псевдо-код, вычисляющий текстурные координаты точки в текстуре, которая впоследствии будет спроецирована на сферу (текстурные координаты находятся в интервале [0, 1]):

```
// поворот Q вокруг оси Y на угол Da
Vec3 Q1 = y axis rotation(Da, Q)
Vec3 Q2 = Q1 / Dr
Real r = SF * (2 * acos(Q2.z) / PI)
Real a = atan(Q2.y / Q2.x) + angle
Vec2 TexCoord = {r * cos(a) / 2.0 + 0.5,
                 r * sin (a) / 2.0 + 0.5 }
```

Листинг 2 Трассировка лучей

4 Алгоритм работы с виртуальным экраном

Идея преобразования видео, предназначенного для просмотра на плоском экране, к fisheye-проекции состоит в использовании так называемого виртуального экрана. Составляется трехмерная сцена, состоящая из экрана, на котором отображается исходное видео. Экран может быть плоским, либо цилиндрическим и находиться в различных позициях (перед зрителями, сбоку, сверху и т. п.). Можно объединить несколько экранов в одной сцене для одновременного отображения нескольких видео файлов. После составления сцена проецируется на плоский кадр с использованием fisheye-камеры (Рис. 4). Затем полученное fisheye-видео будет проецироваться на сферический экран с помощью описанных выше методов.

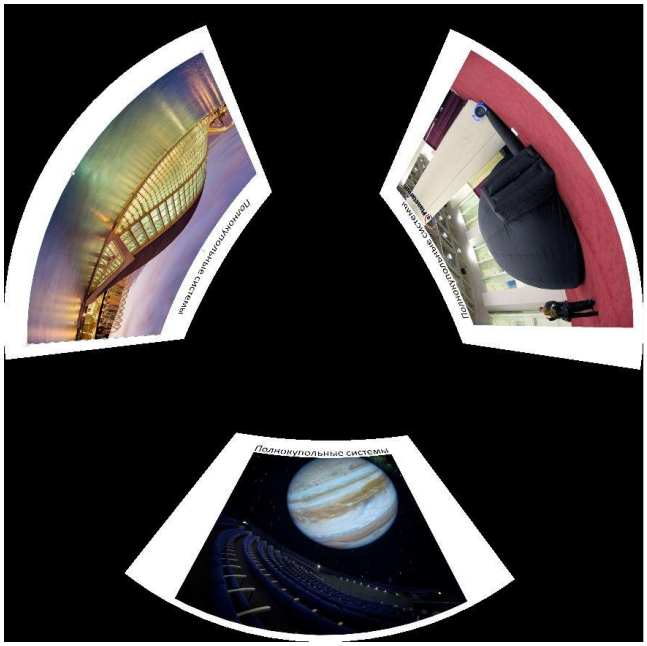

Рис. 4 Fisheye-кадр с виртуальными экранами

В отличии от обычной камеры, для которой проецирование выражается простым умножением трехмерных координат на проекционную матрицу, проецирование для fisheye-камеры не является аффинным преобразованием координат. Вместо этого преобразование выражается алгоритмически с использованием шейдеров (Листинг 3). Так как fisheye-проекция не зависит от параметров полнокупольной системы (единственный параметр проекции - разрешение результирующего изображения), то данное преобразование можно выполнять заранее.

```
vec3 fisheye(vec3 v)
{
  vec3 win;

  float view scaling factor = 1.0 / Fov * 180.0 /
              PI * min(Viewport[2], Viewport[3]);


  float oneoverh = 1.0 / sqrt(v.x * v.x +
                   v.y * v.y);
  float a = 0.5 * PI + atan(v.z * oneoverh);
  if (a > 0.5 * PI)
    a = 0.25 * (PI * PI) / (PI - a);
  float f = (a * view_scaling_factor) * oneoverh;

  win.x = Viewport[0] + Viewport[2] / 2.0 +
          v.x * f;
  win.y = Viewport[1] + Viewport[3] / 2.0 +
          v.y * f;
  win.z = (-v.z - zNear) / (zFar - zNear);
  win.x = 2.0 * (win.x - Viewport[0]) /
          Viewport[2] - 1.0;
  win.y = 2.0 * (win.y - Viewport[1]) /
          Viewport[3] - 1.0;
  win.z = 2.0 * win.z - 1.0;
  return win;
}
```

Листинг 3 Преобразование координат в fisheye - проекции

5 ДЕТАЛИ РЕАЛИЗАЦИИ ПРОГРАММЫ ДЛЯ КАЛИБРОВКИ

Программа калибровки позволяет настроить перечисленные выше параметры алгоритма.Процедура непосредственного вычисления карты преобразования путем трассировки лучей реализована на шейдерах OpenGL. Благодаря этому, вычисления выполняются в реальном времени.

Для реализации программы калибровки была выбрана библиотека Qt - кроссплатформенная, свободная (существует некоммерческая версия) библиотека для языка C++. Одним из плюсов библиотеки Qt является встроенная поддержка OpenGL, что упростило разработку приложения.

6 СОЗДАНИЕ ПЛАГИНА ДЛЯ VLC

VLC является продуктом с открытым исходным кодом. Заключительным этапом разработки является добавление необходимой функциональности в VLC посредством создания плагина. Для построения плеера из исходного кода используются make-файлы с надстройками из automake. Для включения в эту систему плагина необходимо добавить файлы с исходными кодами и объявить в них функции инициализации и деинициализации, предоставляемые библиотеками VLC.

Большую часть производимых операций решено было вынести из кода плагина и сгруппировать в специализированной библиотеке. Она содержит в себе функции инициализации и запуска необходимого фильтра, на вход которого подается буфер с исходным изображением и буфер, в котором будет содержаться результирующее изображение. После чего производится расчет карт смещения

пикселов с помощью загруженных из конфигурационного файла параметров и вышеизложенного алгоритма. Стоит отметить, что создание подобной карты проводится однократно, при запуске приложения. Далее используется уже готовая, хранящаяся в памяти, карта смещения. Благодаря этому удается достичь производительности в 60 кадров в секунду.

7 РЕЗУЛЬТАТЫ

В результате проведенных работ был реализован программный комплекс, состоящий из программы калибровки оборудования и плагина для видео плеера, позволяющих проигрывать fisheye-видео на малых полнокупольных системах, использующих проекцию сферического зеркала (Рис. 5). Данный программный комплекс был протестирован на малом куполе Нижегородского Планетария. Планируется использование разработанной системы для демонстрации научно-популярных фильмов.

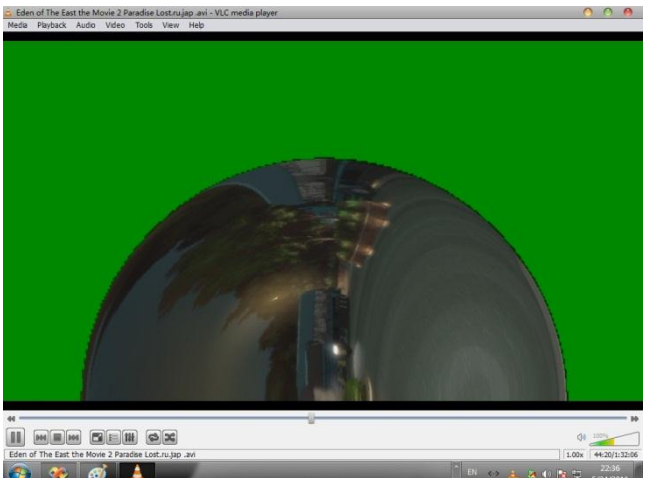

Рис. 5 Работа плагина для VLC

Также был разработан алгоритм и тестовая программа для показа на куполе видео, предназначенного для плоских экранов, путем проецирования его на виртуальный экран.

8 СПИСОК ЛИТЕРАТУРЫ

1. Роджерс Д, Адамс Дж. Математические основы машинной графики / Пер. с англ. – М.: Мир, 2001. – 604с.

2. Рост Р. Дж. OpenGL. Трехмерная графика и язык программирования шейдеров. Для профессионалов – СПб.: Питер, 2005. – 428 с.

3. Paul Bourke. Spherical mirror (Mirrordome) - A new approach to hemispherical dome projection. Planetarian, Vol 34(4), December 2005, pp 5-9.

4. P. D. Bourke. Using a spherical mirror for projection into immersive environments (Mirrordome). Graphite (ACM Siggraph), Dunedin Nov/Dec 2005. Proceedings of the 3rd international conference on Computer graphics and interactive techniques in Australasia and South East Asia. pp 281-284.