|  |
| --- |
| Слайд 2  **Целью** данной работы является разработка программного обеспечения для визуализации шахматных фигур на шахматной доске.  Для достижения поставленной цели требуется решить следующие **задачи**:  1) формализовать задачу;  2) выбрать алгоритм построения реалистичного изображения;  3) разработать функциональную модель программного обеспечения;  4) выбрать средства реализации и реализовать программное обеспечение для визуализации сцены;  5) исследовать зависимость времени генерации кадра от числа полигонов на сцене для варьируемого числа рабочих потоков. |
| Слайд 3  Для визуализации сцены в приложении должны быть заданы: **источник света,** который задается в программе константанами, **камера**, она же наблюдатель, которая определяется вектором взгляда и положением в пространстве и модели фигур. На выходе мы получаем кадр. |
| Слайд 4  На сцене могут быть изображены:  - шахматная доска, 8х8 клеток, с деревянным основанием, клетками черного и белого цветов  - шахматные фигуры, представленные в двух цветах.  Форма и размер шахматных фигур и шахматной доски соответствует **стандарту шахматного оборудования и игровых площадок, предназначенных для проведения турниров Международной шахматной федерации (ФИДЕ**, Fédération Internationale des Échecs). Пример шахматных фигур, приведенный в качестве возможно используемых моделей в соревнованиях ФИДЕ. Данные фигуры оформлены в стиле **Стаунтон**, которые рекомендуется для применения. |
| Слайд 5  В программе была использована глобальная модель освещения. Она состоит из интенсивности рассеянного диффузного и зеркально отраженного света, а так дополнительно учитывает интенсивность света приходящего от луча преломления и луча отражения  Она учитывает взаимное расположение объектов. |
| Слайд 6  Одним из **этапов построении реалистичного изображения** является удаление невидимых линий. Сравнение алгоритмов, проводилась по критериям возможности визуализации отражающих поверхностей, описания произвольных объектов, и ограничения точности изображения разрешающей способностью экрана  Для генерации кадра был выбран алгоритм обратной трассировки лучей в связи с целями задачи. И так как этот метод, для закраски сцены и создания теней не требует дополнительных алгоритмов, он также является и решением задачи построения реалистичного изображения. |
| Слайд 7  Для генерации кадра на основе информации о положении в пространстве объектов сцены, интенсивности источника света, вектора направления камеры и множестве полигонов,  которыми заданы модели фигур использовался алгоритм обратной трассировки лучей, который заключается в испускании луча для каждого пикселя и вычислении цвета этого пикселя |
| Слайд 8  Для поданного на вход луча определяется его пересечение с ближайшим видимым объектом сцены, в случае отсутствия пересечения, пиксель, определяемый лучом, закрашивается цветом фона. Если пересечение найдено, запускается луч, направленный на источник света для определения освещенности точки в пространстве, и запускается луч отражения если поверхность  пересечения является глянцевой. |
| Слайд 9  Для расчета цвета пикселя на основе пересечения луча с ближайшим полигоном сцены, учитывается интенсивность рассеянного света, диффузное отражение, которое определяется вектором нормали и вектором направленным на источник, зеркальным отражением, в случае если поверхность является глянцевой, которое определяется вектором отраженного луча и вектором взгляда, а так же испускается вектор отражения и для него производятся такие же действия, рекурсивно. Рекурсия в данном алгоритма ограничена |
| Слайд 10  В качестве средств реализации был выбрал язык С++, модульное тестирование было реализовано с помощью фреймворка GoogleTest и покрытие кода вычислялось с помощью утилиты gcov. |
| Слайд 11  При разработке программного обеспечения использовался объектно-ориентированный подход и паттерны проектирования, для улучшения декомпозиции задачи и облегчения модификаций кода.  При реализации программного обеспечения использовались такие паттерны как **фасад** (FacadeScene), **стратегия** (TransformAction), **команда**, он представлен здесь одним базовым классом BaseCommand, его реализации были вынесены с целью сохранения читабельности диаграммы, так же представлены паттерны **директор** и **компоновщик**, для создания классов моделей фигур. |
| Слайд 12  Программа была покрыта модульными тестами. Были созданы наборы тестов для основных вычислительных конструкций, для методов класса Camera, RayTracing, Ray, Triangle.  В качестве метрики успешности модульного тестирования было выбрано покрытие кода. Были покрыты тестами 35% процентов строк кода. |
| Слайд 13  Было проведено функциональное тестирование, которое заключалось в рассмотрении нескольких положений камеры для различной расстановки фигур относительно друг друга при использовании глянцевой и матовой поверхности шахматной доски. Вы видите наиболее показательные ракурсы. |
| Слайд 14  В качестве исследования была выбрана зависимости времени генерации кадра от числа полигонов на сцене для варьируемого числа рабочих потоков.  По результатам тестирования были сделаны выводы что при количестве полигонов на сцене большем 200, наиболее эффективными для машины на которой производились тесты является использования 32 потоков.  поскольку количество полигонов, которыми задаются модели шахматных фигур определяет точность их визуального восприятия. |
| Слайд 15  Было разработано программное обеспечение, которое позволяет получить реалистичное изображение шахматных фигур на шахматной доске.  Все поставленные задачи были решены, и поставленная цель достигнута. Спасибо за внимание, готова продемонстрировать ПО. |