Независимый электронный журнал о разработке компьютерных игр

FPS

№14
2011

.:.:.: Blender :: OpenCV :: Язык D :: GLSL
...и многое другое

Нас читают:

gcup.ru
Все для начинающего и профессионального разработчика игр

xtreme3d.narod.ru
Сайт движка Xtreme3D

make-games.ru
Портал создания игр

gamer-club.ucoz.com
Все для создания игр без программирования и не только

mizzystic.ru
Крупнейший информационный Game Maker портал

xbobr.at.ua
Создание игр, программ, музыки

portalgame.net.ru
Игровой портал

gamesfpscreator.at.ua
Сайт для помещений игр и обсуждаловок

dapf.us
Design And Programming Forum

gameshaker.ukoz.ru
Все для редактирования и создания игр

esate.ru
Мультимедиа-сообщество

www.mobkiosk.com
"Мобильный киоск"

dogames.ru
Мастерская игр

gamecreate.ru
Создай свою игру!

igrostroenie.net.ru
Игровые движки и ресурсы

rus.game-maker.ru
Русское сообщество Game Maker

http://gacon.ucoz.ru
Сайт, посвященный разработке игр

http://game.oxnull.net
Разработка игр на конструкторе Game Maker

Главный редактор: Тимур Гафаров
Дизайн и верстка: Тимур Гафаров
По вопросам сотрудничества обращаться по адресу clocktower89@mail.ru или gecko0307@gmail.com.
Официальный сайт журнала: http://fps-magazine.narod.ru

ЭЛЕКТРОННЫЙ ЖУРНАЛ о разработке компьютерных игр

№ 14'11

В этом выпуске:

Blender

Перспективы развития

13 апреля, после четырех лет разработки, наконец увидел свет очередной стабильный релиз Blender 2.57. В июне этого года планируется выпустить Blender 2.58, в котором будут исправлены ошибки и доведены до конца некоторые возможности (например, отображение нескольких сцен в разных окнах). После этого разработчики переключатся на развитие новой экспериментальной ветки 2.6.

Blender 2.57 уже себе несет в себе громадное количество концептуальных инноваций и багфиксов, однако впереди нас ждет еще немало интересного. Как отмечают сами разработчики, хоть эта версия и считается стабильной, ее трудно назвать полностью завершенной – ведь в новой ветке пока не реализованы некоторые возможности 2.49. Итак, чего же пользователям ожидать в ближайшем будущем?

Вполне вероятно, что после релиза 2.58 в основную ветку внесут Bmesh – новую систему работы с полигональной сеткой, поддерживающую N-гоны (поверхности с произвольным числом ребер). Это, в свою очередь, ускорит интеграцию других интересных экспериментальных разработок, для которых Bmesh является критичной необходимостью. Среди них стоит отметить Unlimited Clay – инструмент воксельной лепки, позволяющий достигать неограниченной детализации, не беспокоясь о растяжении полигонов. Аналогичная возможность, кстати, присутствует в программе Sculptris от разработчиков знаменитого ZBrush.

Брехт ван Ломмель, один из разработчиков Октан (Octane Renderer), вернулся в Blender Foundation для работы по совершенствованию встроенного рендера Blender Internal. Перед началом работы над фильмом «Синтел», Брехт уже пытался реорганизовать и улучшить код, но столкнулся с трудно преодолимыми проблемами при попытке реализации в существующем рендере некоторых возможностей. В связи с этим, он намерен разработать новый движок рендеринга и освещения на основе узлов, который полностью заменит BI. Возможно, это произойдет после проекта Mango.

Кстати, о последнем. Как заявил Тон Роозендаал, в рамках Mango будет снято что-то вроде демонстрации композитинга и VFX в Blender – короткометражный фильм с элементами трехмерной графики. Blender Foundation также планирует другой крупный проект под кодовым названием Gooseberry, работа над которым назначена на 2012-2014 гг.

Кроме того, недавно была анонсирована «Gemini Rising» – первая голливудская картина, в которой основным инструментом для создания спецэффектов будет Blender! Это научно-фантастический фильм об экипаже NASA, который обнаружил заброшенный инопланетный космический корабль, блуждающий в космосе. Все игровые сцены уже отсняты, на сайте проекта (www.geminirisingfilm.com) публикуются задания для VFX-художников.

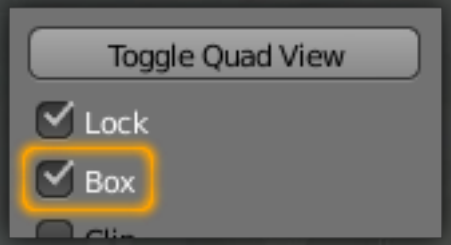
Blender 2.5

Полезные советы

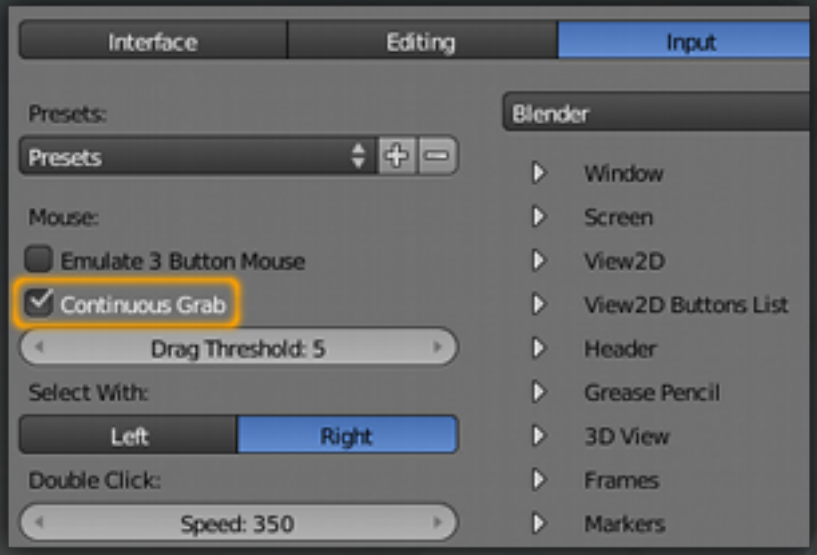
● Чтобы поместить 3D-курсор в центр координатной системы (в точку 0,0,0), нажмите комбинацию Ctrl+C.

● В режиме Quad View (View > Toggle Quad View) по умолчанию окно пользовательского вида (User) отображает ортогональную проекцию. Ее можно переключить на перспективную клавишей Numpad 5.

● В том же режиме Quad View можно синхронизировать масштаб и перемещение точки обзора в трех ортогональных проекциях (Top, Front, Right). Для этого активизируйте панель свойств (клавиша N) и на вкладке Display поставьте галочку в опции Box. Напомним, что перемещать точку обзора можно средней клавишей мыши с одновременно зажатой клавишей Shift.



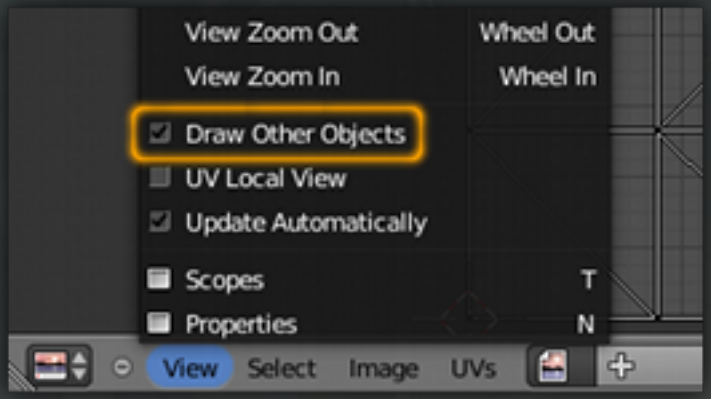
● Если по умолчанию курсор мыши не «перескакивает» на противоположную сторону экрана при масштабировании, перемещении и других «мышиных» операциях, зайдите в настройки (User Preferences) и активируйте опцию Continuous Grab в секции Input.



● Наведите курсор мыши на любое поле ввода значения и нажмите I - значение будет сохранено в текущем ключевом кадре.

● Кнопка "Запись" на панели инструментов временной шкалы активизирует автоматическое добавление ключевых кадров при изменении параметров объектов.

● Опция View > Draw Other Objects в редакторе UV/изображений включает режим отображения разверток других выделенных объектов, использующих текущую текстуру. Это очень полезно, если вы создаете развертки нескольких мешей с учетом того, что они будут использовать одну и ту же текстуру. Например, текстура автомобиля может содержать изображения одновременно и для кузова, и для колес.



● Иногда бывает так, что Blender с самого начала работает слишком медленно. В этом случае попробуйте зайти в настройки (User Preferences) и в секции System переключить опцию Window Draw Method с Full на Overlap или Overlap Flip.

● В Blender 2.5 клавиша "пробел" не вызывает меню добавления примитива. Эту функцию теперь выполняет комбинация Shift+A.

● После смены любых настроек можно сохранить их для использования по умолчанию. Для этого нажмите Ctrl+U.

«Талисманы» компьютерной графики

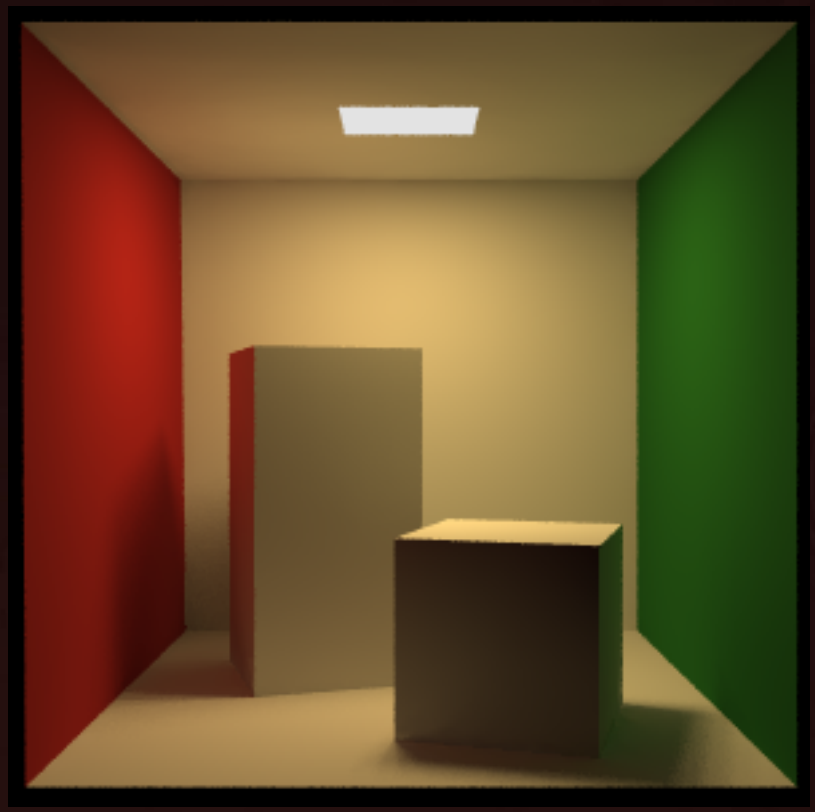
Мы продолжаем наш обзор «талисманов» компьютерной графики, начатый в «FPS» #11 ('10). Если у Вас есть что добавить или Вы можете рассказать о других, не упомянутых здесь талисманах, просим писать на почтовый адрес редакции: clocktower89@mail.ru (или лично главному редактору: gecko0307@gmail.com).

Cornell Box

Cornell Box изначально возник как тест для проверки точности рендера путем сравнения полученного изображения с его аналогом из реального мира (то есть, фотографией). При создании модели соблюдаются точные масштабы и фотометрические данные используемой сцены. Тест впервые был применен в Корнелльском университете США (Cindy M. Goral, Kenneth E. Torrance, Donald P. Greenberg, Bennett Battaile, «Modeling the Interaction of Light Between Diffuse Surfaces», SIGGRAPH '84).

Традиционный Cornell Box состоит из «коробки» (красная левая стенка, зеленая правая стенка, белое дно и крышка), одного источника света и нескольких объектов (параллелепипеды, сферы и др.). Зеленая стенка иногда заменяется на синюю. В рендерах с поддержкой глобального освещения (GI) этот тест показывает переотражение света от цветных стенок.

Очень часто Cornell Box используется также для демонстрации преломления и отражения света, каустики, для сравнения различных методов GI и т. д. В наши дни эту модель, как правило, используют только из-за своих визуальных характеристик и уже не сравнивают с реальной фотографией.



Стэнфордский дракон и «Счастливый Будда»

Кроме всем известного кролика, в репозитории трехмерных сканов Стэнфордского университета США можно найти ряд других моделей, которые также приобрели статус «талисманов». В первую очередь, это Стэнфордский дракон (Stanford Dragon) и «Счастливый Будда» (Happy Buddha).

Дракон состоит из 871414 треугольников, полученных путем сканирования статуэтки на Cyberware 3030 Model Shop. Модель была создана в 1996 г. и ныне находится в свободном доступе в форматах PLY, VRML, VL и т. д. «Будда» был получен в том же 1996 г. Брайаном Керлессом и Марком Левоем. Реконструированная детализация – 1.1 млн. треугольников (оригинальная при сканировании – 9.2 млн.). Обе модели широко используются для проверки различных материалов, а также методов рендеринга.

Кроме вышеупомянутых моделей, на сайте Стэнфорда (по ссылке http://graphics.stanford.edu/data/3Dscanrep/) также доступны «Броненосец» (Armadillo), тайская статуэтка и другие сканы. Любопытно, что на странице имеется предупреждение о «неуместном использовании» (inappropriate use) некоторых моделей, представляющих собой религиозные символы или содержащих какие-либо элементы религиозного или культурного значения. Пользователям настоятельно рекомендуется проявить уважение к чувствам верующих: не деформировать модели и не «глумиться» над ними.

В предыдущем номере журнала мы рассмотрели простейшую программу, использующую OpenCV – открытую кроссплатформенную библиотеку компьютерного зрения. OpenCV уже включает в себя базовые средства ввода-вывода, поэтому для создания простого окошка с видеозахватом можно ограничиться только ими. Однако в проектах посложнее, таких как игры, стандартных средств OpenCV становится мало. Чтобы рисовать поверх видео свою собственную графику (в данном случае – через OpenGL), нужен способ «конвертации» кадров видео в текстуру, которая будет натянута на прямоугольник, служащий фоном.

Мы рассматриваем работу с OpenCV в языке D (личное предпочтение автора этих строк) – но он, по крайней мере в этой ситуации, не очень сильно отличается от C и C++, поэтому при желании портировать код будет несложно. Для создания окна и взаимодействия с устройствами ввода мы будем использовать библиотеку SDL (Simple Direct media Library, www.libsdl.org).

```
module main;
import std.stdio;
import std.string;
```

Для работы с OpenGL и SDL можно использовать Derelict:

```
import derelict.sdl.sdl;
import derelict.opengl.gl;
import derelict.opengl.glu;
import derelict.util.compat;
```

Функция создания текстуры из изображения OpenCV (вызывается только один раз в начале программы):

```
int createTexture(IplImage *image, GLuint *texture)
{
   if (!image) return -1;
   glGenTextures(1, texture);
   glBindTexture(GL_TEXTURE_2D, *texture);
   glTexParameteri(GL_TEXTURE_2D,
      GL_TEXTURE_MAG_FILTER, GL_LINEAR);
   glTexParameteri(GL_TEXTURE_2D,
      GL_TEXTURE_MIN_FILTER, GL_LINEAR);
   glPixelStorei(GL_UNPACK_ALIGNMENT, 1);
   glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB,
      image.width, image.height, 0, GL_BGR,
      GL_UNSIGNED_BYTE, image.imageData);
   return 0;
}
```

Функция обновления текстуры (вызывается при смене кадра видео):

```
int updateTexture(IplImage *image, GLuint *texture)
{
   if (!image) return -1;
   glBindTexture(GL_TEXTURE_2D, *texture);
   glTexSubImage2D(GL_TEXTURE_2D, 0, 0, 0,
      image.width, image.height, GL_BGR,
      GL_UNSIGNED_BYTE, image.imageData);
   return 0;
}
```

Функция уничтожения текстуры:

```
void freeTexture(GLuint *texture)
{
   glDeleteTextures(1, texture);
}


int main()
{
```

Линкуемся с библиотеками OpenGL и SDL:

```
DerelictSDL.load();
DerelictGL.load();
DerelictGLU.load();
```

Инициализируем SDL:

```
if (SDL_Init(SDL_INIT_VIDEO) < 0)
   throw new Exception("couldn't init SDL:\n"
      ~ toDString(SDL_GetError()));
```

Выход из SDL можно оформить здесь же, указав область вызова:

```
scope (exit)
{
   if (SDL_Quit != null)
      SDL_Quit();
}
```

Устанавливаем заголовок окна и атрибуты OpenGL:

```
SDL_WM_SetCaption("OpenCV + OpenGL Demo", "OpenCV");
SDL_GL_SetAttribute(SDL_GL_BUFFER_SIZE, 32);
SDL_GL_SetAttribute(SDL_GL_DEPTH_SIZE, 16);
SDL_GL_SetAttribute(SDL_GL_DOUBLEBUFFER, 1);
```

Задаем видеорежим:

```
int width = 640;
int height = 480;
if (SDL_SetVideoMode(width, height, 0, SDL_OPENGL)==null)
   throw new Exception("failed to set video mode:\n"
      ~ toDString(SDL_GetError()));
```

Запрашиваем доступные расширения OpenGL:

```
DerelictGL.loadExtensions();
```

Создаем вид:

```
glClearColor(0.0, 0.0, 0.0, 1.0);
glViewport (0, 0, cast(GLsizei)width,
                  cast(GLsizei)height);
```

Переменная texture будет хранить текстуру текущего кадра с камеры:

```
GLuint texture;
```

Организуем захват с веб-камеры:

```
CvCapture *capture = null;
capture = cvCreateCameraCapture(CV_CAP_ANY);
if (!capture)
   throw new Exception("couldn't initialize webcam");
```

Запрашиваем у камеры первый кадр и создаем текстуру:

```
IplImage *frame;
frame = cvQueryFrame(capture);
if (!frame) return 1;
freeTexture(&texture);
createTexture(frame, &texture);
```

Служебная переменная angle для хранения поворота объекта:

```
GLfloat angle = 0.0;
```

Начинаем основной цикл:

```
uint counter = 0;
bool running = true;
while(running)
{
```

Завершаем цикл, если пользователь закрыл окно или нажал клавишу Escape:

```
SDL_Event event;
while (SDL_PollEvent(&event))
{
   switch (event.type)
   {
      case SDL_KEYDOWN:
         if (SDLK_ESCAPE == event.key.keysym.sym)
            running = false;
         break;
      case SDL_QUIT:
         running = false;
         break;
      default:
         break;
   }
}
```

Оптимизации ради будем захватывать очередной кадр только через каждые 20 шагов:

```
if (counter==20) {
   frame = cvQueryFrame(capture);
   if (!frame) break;
   updateTexture(frame, &texture);
   counter=0;
}
else counter += 1;
```

Устанавливаем ортографическую проекцию и рисуем видео:

```
glClear(GL_COLOR_BUFFER_BIT);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glOrtho(0, width, height, 0, -1, 1);
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glTranslatef(width/2, height/2, 0);
glScalef(width/2, height/2, 1);
glEnable( GL_TEXTURE_2D );
glBindTexture( GL_TEXTURE_2D, texture );
glColor4f(1.0f, 1.0f, 1.0f, 1.0f);
glBegin (GL_QUADS);
   glTexCoord2d(1.0, 0.0); glVertex2d(-1.0, -1.0);
   glTexCoord2d(0.0, 0.0); glVertex2d(+1.0, -1.0);
   glTexCoord2d(0.0, 1.0); glVertex2d(+1.0, +1.0);
   glTexCoord2d(1.0, 1.0); glVertex2d(-1.0, +1.0);
glEnd();
```

Возвращаем перспективную проекцию (конечно, это надо было лучше сделать через glPushMatrix и glPopMatrix, но так нагляднее):

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluPerspective(60,
   cast(GLfloat)width / cast(GLfloat)height, 1.0, 100.0);
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
```

Рисуем трехмерный объект (вращающуюся плоскость) поверх видео. Ради интереса можно натянуть на нее нашу текстуру с кадром видео – ведь ее все еще можно использовать.

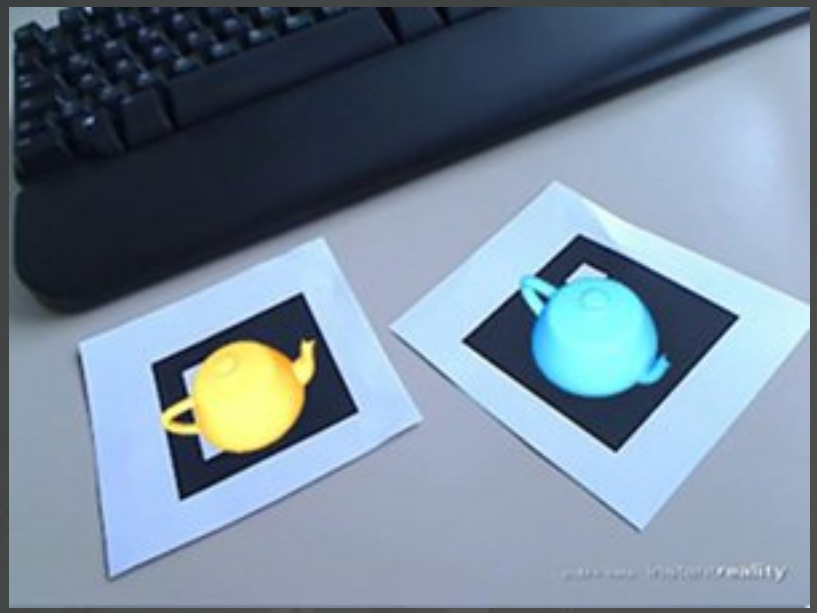
```
glTranslatef(0, 0, -5);
glRotatef(angle, 1.0f, 1.0f, 1.0f);
glBindTexture(GL_TEXTURE_2D, texture);
glColor4f(1.0f, 0.0f, 0.0f, 0.5f);

glBegin(GL_QUADS);
   glTexCoord2d(1.0, 0.0); glVertex2d(-1.0, -1.0);
   glTexCoord2d(0.0, 0.0); glVertex2d(+1.0, -1.0);
   glTexCoord2d(0.0, 1.0); glVertex2d(+1.0, +1.0);
   glTexCoord2d(1.0, 1.0); glVertex2d(-1.0, +1.0);
glEnd();

glDisable( GL_TEXTURE_2D );
angle += 0.5;
```

Обновляем буферы и ждем 1 миллисекунду, чтобы разгрузить CPU под другие процессы:

```
SDL_GL_SwapBuffers();
SDL_Delay(1);
}
```

Освобождаем устройство захвата, уничтожаем текстуру и завершаем работу программы:

```
cvReleaseCapture(&capture);
freeTexture(&texture);
return 0;
}
```



Уолтер Брайт, автор языка D и глава компании Digital Mars, недавно объявил о начале конкурса, главным призом которого станет новенький iPad 2 с WiFi и 16 Гб памяти. Для участия необходимо написать статью о D и отправить ее в группу новостей digitalmars.d.

Напиши статью о D и выйграй iPad 2!

Статьи принимаются до 1 июня 2011 г. Победитель будет определен голосованием участников группы, которое продлится **до 8 июня.** Правила конкурса следующие:

1. Статьи принимаются и распространяются по лицензии Creative Commons;

2. Участник предоставляет Digital Mars право передачи статей в издательства, такие как DDJ and Artima.com;

3. В этом случае участник может получить дополнительный гонорар;

4. Участник предоставляет Digital Mars право размещать статьи на сайтах компании;

5. Участник может размещать черновики статей, обновленные редакции, учитывать читательские отклики при редактировании.

6. Предмет статьи – язык программирования D;

7. Каждый участник может разместить любое количество статей;

8. Участвовать в конкурсе может любой желающий;

9. Законченные варианты статей отправляются в группу новостей digitalmars.d. В качестве темы следует указывать «Submission»;

10. Победитель по желанию может получить приз в денежном эквиваленте.

Желаем удачи!

Язык D. Практикум

В этой статье я хочу продемонстрировать пару нехитрых приемов, которые использую в своих проектах на D.

Практически в любой игре (особенно в двумерной) необходимо хранить поле каких-либо дискретных значений. Это может быть карта тайлов, карта игровых зон или блоков для проверки столкновений. Можно использовать для этой цели большой зарезервированный массив – но если координаты выйдут за его пределы, придется его динамически расширять, а это не самая быстрая процедура. Если учесть, что собственно значения хранятся не во всех ячейках поля, и большинство их все равно будет заполнено нулями, можно взять за основу ассоциативный массив.

Поэтому я предлагаю шаблон класса Map, который будет служить оберткой над двумерным ассоциативным массивом:

```
class Map(T) {
   protected:
   T[int][int] hash;
   T defaultValue;
   int min_x = 0;
   int min_y = 0;
   int max_x = 0;
   int max_y = 0;
   public:
   this(T fill) {
       defaultValue = fill;
   }
   T opIndex(int x, int y) {
       if (x in hash)
       {
           if (y in hash[x]) return hash[x][y];
           else return defaultValue;
       }
       else return defaultValue;
   }
   void opIndexAssign(T val, int x, int y) {
      if (x<min_x) min_x = x;
      else if (x>max_x) max_x = x;
      if (y<min_y) min_y = y;
      else if (y>max_y) max_y = y;
      hash[x][y] = val;
   }
}
```

Благодаря перегрузке операторов opIndex и opIndexAssign, к нему можно обращаться как к некоему виртуальному массиву:

```
auto map = new Map!int(0);
map[0,0] = 5;
map[10,4] = 2;
```

Теперь проверим значения:

```
writeln(map[0,0],", ",map[10,4]);
```

Если все правильно, программа выведет 5, 2. А если обратится к несуществующей ячейке? Например, map[999,55]. Это вполне безопасно, так как класс уже готов к такому повороту событий – оператор индексирования просто вернет значение по умолчанию (0), которое мы передали в конструктор класса.

Конечно, шаблон Map нельзя назвать идеальным – необходимо еще добавить перегрузку арифметических операций, а также методы-диапазоны для итерирования значений массива. Оставляю эту задачу читателю для самостоятельного решения.

Во многих стратегиях и симуляторах используется интерфейс, который можно охарактеризовать как «screen to world» (из экранных координат – в трехмерные). Игрок использует курсор мыши для управления трехмерными объектами на плоскости игрового поля. Классический пример – режим покупки и строительства в играх серии The Sims.

Функция, которая при этом используется, должна выполнить преобразование координат точки на плоскости экрана в координаты соответствующей точки на нулевой плоскости трехмерного пространства (XZ). Приведу простейшую реализацию такого преобразования:

```
vector2f ScreenToWorld(
        float ScreenX,
        float ScreenY,
        float WindowWidth,
        float WindowHeight,
        vector3f CameraPosition,
        vector3f CameraForwardVector,
        vector3f CameraUpVector,
        bool Snap = false,
        float FOV = 60.0f)
{
   float aspectRatio = WindowWidth/WindowHeight;
   float tFOV = tan(FOV * PI / 360.0f);
   CameraUpVector = CameraUpVector * tFOV;
   vector3f v = cross(CameraUpVector, CameraForwardVector);
   v *= aspectRatio;

   float screenx = 1.0f-2.0f*(ScreenX)/WindowWidth;
   float screeny = 1.0f-2.0f*
                   (WindowHeight-ScreenY)/WindowHeight;

   float mx = CameraForwardVector.x +
              CameraUpVector.x * screeny + v.x * screenx;
   float my = CameraForwardVector.y +
              CameraUpVector.y * screeny + v.y * screenx;
```

```
    float mz = CameraForwardVector.z + CameraUpVector.z * screeny + v.z * screenx;

    float WorldX = Snap?
          floor(CameraPosition.x - mx * CameraPosition.y / my) : (CameraPosition.x - mx * CameraPosition.y / my);
    float WorldY = Snap?
          floor(CameraPosition.z - mz * CameraPosition.y / my) : (CameraPosition.x - mx * CameraPosition.y / my);
    return vector2f(WorldX,WorldY);
}
```

Аргументы функции:

float ScreenX — координата X на экране (относительно левого верхнего угла окна);
float ScreenY — координата Y на экране (относительно левого верхнего угла окна);

float WindowWidth – ширина окна;
float WindowHeight – высота окна;

vector3f CameraPosition – абсолютные координаты камеры в трехмерном пространстве;
vector3f CameraForwardVector – абсолютный вектор направления камеры;
vector3f CameraUpVector – абсолютный вектор Up камеры (вектор, направленный вверх;
он меняется при вращении камеры вокруг вектора направления);

bool Snap – в некоторых играх (в том же Sims, например) объекты создаются строго дискретно, т. е. выравниваются по сетке. Эта опция позволяет включить выравнивание по сетке с единичным шагом (1x1);

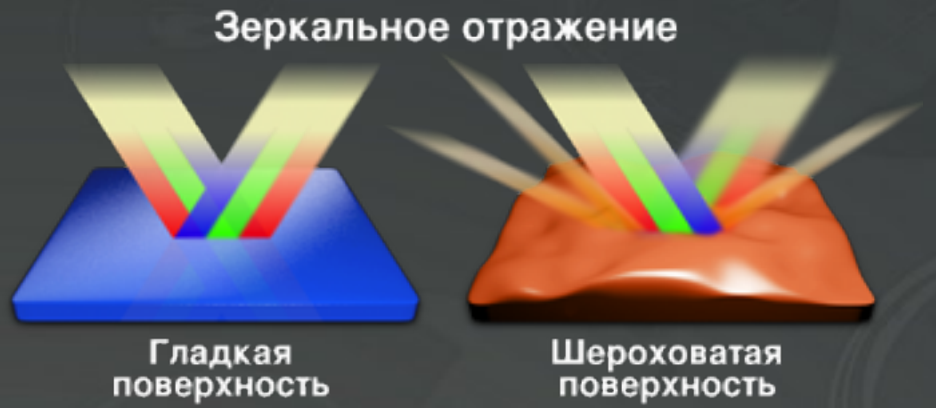
float FOV – сокращение от field of view, угол зрения камеры.

Модели освещения

Бликовая составляющая. Распределение Бекмана

Бликовая составляющая (specular term) – это количество света, зеркально отраженного поверхностью. Блик появляется лишь в тех участках поверхности, нормаль в которых совпадает с половинным вектором (half-angle) между направлением на источник света и направлением наблюдателя. Таким образом, блик – не что иное, как изображение источника света, отраженное на поверхности объекта. Однако, если учесть, что в компьютерной графике используются идеализированные объекты, возникает закономерный вопрос: почему точечный источник света, не имеющий объема и невидимый сам по себе, отражается как относительно крупный размытый световой блик? Это объясняется наличием микрограней. Считается, что поверхности не идеально гладкие, но состоят из множества крошечных граней, каждая из которых является идеальным зеркальным отражателем. Иными словами, любая поверхность в той или иной степени шероховата. Микрограни имеют собственные вектора нормалей, отклонение которых от основной нормали поверхности соответствует параметру шероховатости поверхности.

Ближе к центру блика, где нормаль поверхности близка к половинному вектору, количество микрограней, также отражающих в направлении половинного вектора, достаточно велико – и блик имеет наибольшую интенсивность. Чем дальше мы удаляемся от центра блика, тем больше разница между нормалью и половинным вектором, и меньше количество зеркальных отражений от микрограней – блик постепенно теряет яркость. Следовательно, параметр шероховатости для бликовой составляющей будет определять степень «размытости» блика.



Цвет блика может не совпадать с цветом материала. Это справедливо для некоторых многослойных материалов – например, пластик представляет собой «слоеный пирог» из пигмента и прозрачного полимера. Блик дают прозрачные слои, а диффузное рассеивание – цветные. Однородные материалы такого эффекта не производят, и блики на них имеют тот же цвет, что и сами материалы. Яркий пример – металл.

Существует довольно много моделей освещения, предусматривающих микрограни. Большинство из них допускают равномерное распределение нормалей микрограней относительно нормали поверхности; эти модели называются изотропными. Если микрограни распределяются с предпочтением какого-либо заданного направления вдоль поверхности, то такая модель анизотропна.

Наиболее распространенной изотропной моделью, учитывающей микрограни, является распределение Бекмана. В отличие от эмпирических моделей Фонга и Блинна, она физически корректна.

Формула Бекмана имеет следующий вид:

$$k_{spec} = \frac{exp(-tan^2\,\alpha\,/m^2)}{4m^2 cos^4\,\alpha},\ \alpha = arccos(N \cdot H)$$

где k_{spec} – коэффициент зеркального отражения, m – параметр шероховатости, N – нормаль поверхности, H – половинный вектор.

Поскольку

$$tan^2\,\alpha\,/m^2 = \frac{1-cos^2\,\alpha}{cos^2\,\alpha\,m^2}$$

то формулу можно выразить так:

$$k_{spec} = \frac{exp\left(\frac{(N\cdot H)^2-1}{(N\cdot H)^2 m^2}\right)}{4m^2(N\cdot H)^4} = \frac{exp\left(\frac{(N\cdot H)^2-1}{(N\cdot H)^2 m^2}\right)\frac{1}{(N\cdot H)^2 m^2}}{4(N\cdot H)^2}$$

На практике распределение Бекмана часто используют в более сложных моделях (например, в модели Кука-Торренса). Модели с учетом микрограней позволяют наиболее реалистично передавать фактуру различных тканей (бархата, вельвета, шелка) и органических поверхностей.

Реализация на GLSL*

Вершинная программа:

```
varying vec4 V_eye;
varying vec4 L_eye;
varying vec4 N_eye;

void main(void)
{
   gl_Position = ftransform();
   V_eye = gl_ModelViewMatrix * gl_Vertex;
   L_eye = gl_LightSource[0].position - V_eye;
   N_eye = vec4(gl_NormalMatrix * gl_Normal, 1.0);
   V_eye = -V_eye;
}
```

Фрагментная программа:

```
varying vec4 V_eye;
varying vec4 L_eye;
varying vec4 N_eye;

const float roughness = 1.0;

void main(void)
{
   vec4 Ca = gl_FrontMaterial.ambient;
   vec4 Cd = gl_FrontMaterial.diffuse;
   vec4 Cs = gl_FrontMaterial.specular;

   vec3 V = normalize(vec3(V_eye));
   vec3 L = normalize(vec3(L_eye));
   vec3 N = normalize(vec3(N_eye));
   vec3 H = normalize(L + V);
   float NL = max(0.0, dot(N, L));
   float NH = max(1.0e-7, dot(N, H));

   float diffuse = clamp(NL, 0.0, 1.0);

   float NH_sq = NH * NH;
   float NH_sq_r  = 1.0 / (NH_sq * roughness * roughness);
   float roughness_exp = (NH_sq - 1.0) * ( NH_sq_r );
   float specular = exp(roughness_exp)*NH_sq_r/(4.0*NH_sq);

   gl_FragColor = Ca + Cd*diffuse + Cs*specular;
   gl_FragColor.a = 1.0;
}
```

** - предполагается, что в приложении уже включены и настроены соответствующие параметры OpenGL (позиция источника света, свойства материала и др.) Приведенная реализация в целях упрощения не учитывает интенсивность источника света. Исходный код предоставлен как общественное достояние (Public Domain) и может быть использован безо всяких ограничений.*

Шейдеры на все случаи жизни

Эффект «ударной волны» на GLSL

Помимо HDR и Glow, шейдеры позволяют создавать огромное количество эффектов постобработки, связанных с деформацией изображения. Это и подводное искажение, и тепловое, и эффекты стекла и невидимости как в «Хищнике», и многое другое. Отдельный интерес представляет эффект с расходящимися от центра концентрическими волнами. Его можно использовать, например, для эффектов «искажения пространства» во время действия какого-либо фантастического оружия или двигателя.

Вершинная программа:

```
vec2 uv;
void main(void)
{
   gl_Position = ftransform();
   uv = gl_MultiTexCoord0.xy;
}
```

Во фрагментную программу передается текстура, в которую заранее отрендерена сцена (это можно сделать через FBO или glCopyTexImage2D). Кроме того, передаются параметры center (координаты центра эффекта на экране), timeElapsed (время в секундах, прошедшее с начала действия эффекта) и alpha (прозрачность).

```
uniform sampler2D gl_Texture0;
varying vec2 uv;
uniform vec2 center;
uniform float timeElapsed;
uniform float alpha;
const vec3 shockParams = vec3(10.0,0.8,0.1);

void main(void)
{
   vec2 texCoord = uv;
   float distance = distance(uv, center.xy);
   if ( (distance <= (timeElapsed + shockParams.z)) &&
        (distance >= (timeElapsed - shockParams.z)) )
   {
       float diff = (distance - timeElapsed);
       float powDiff = 1.0 - pow(abs(diff*shockParams.x),
              shockParams.y);
       float diffTime = diff  * powDiff;
       vec2 diffUV = normalize(uv - center.xy);
       texCoord = uv + (diffUV * diffTime);
   }
   gl_FragColor = texture2D(gl_Texture0, texCoord);
   gl_FragColor.a = alpha;
}
```

«Искажение пространства» в игре Prey

Патенты в игровой индустрии

Патент — это способ закрепления за патентодержателем набора исключительных прав, позволяющих ему запрещать кому бы то ни было воспроизведение, использование, продажу, выдвижение на продажу и импорт запатентованного изобретения. Конкретные условия действия патентов и конкретные наборы исключительных прав могут сильно меняться от государства к государству, поэтому действие патента строго ограничено юрисдикцией той страны, где он был выдан. Если обладатель патента хочет добиться защиты изобретения в нескольких юрисдикциях, ему нужно для каждой из них оформить отдельную патентную заявку.

Патенты на программное обеспечение (или «софтверные патенты») — специфическая разновидность патентов, действительная в некоторых странах мира, прежде всего в США. Софтверные патенты — одна из наиболее обсуждаемых тем в области информационных технологий.

Патенты на ПО — это монополия на идею. Даже если кто-нибудь придет к идее самостоятельно, он не сможет ею воспользоваться, если она уже кем-то была запатентована. Более того, в США при разработке ПО практически невозможно не нарушить патент (известно, что та же Microsoft владеет самыми абсурдными патентами, вплоть до "запуска приложений под учетной записью администратора"). Даже если вы, что закономерно, захотите узнать, какие патенты потенциально нарушает ваша программа, перед тем как ее написать, – вам это не удастся. Многие патенты представляют собой сектретную информацию и не подлежат публикации какое-то время – в течение которого вполне реально не только написать программу, но и успешно распространить ее. Именно такая ситуация в свое время возникла с патентом на алгоритм сжатия LZW.

Сторонники софтверных патентов настаивают на том, что этот механизм защиты исключительных прав на программы для ЭВМ необходим для стимулирования инновационной деятельности. Настораживает, однако, что этот аргумент выдвигают в основном крупные компании. В США, где на рынке информационных технологий главную роль играют как раз крупные фирмы, уже сложились определенные правила игры между обладателями солидных патентных портфелей, позволяющие им поддерживать равновесие сил «в сво м кругу» и успешно противостоять менее крупным компаниям. При получении патентного иска крупная фирма может прибегнуть к проверенному способу подачи встречного патентного иска, и если число полученных софтверных патентов у двух участников тяжбы сопоставимо, то дело рано или поздно будет улажено полюбовно.

На сегодня в Европе, как и в РФ, софтверные патенты законодательно не существуют. Однако многие специалисты в области игровой индустрии сейчас стремятся в США в надежде найти высокооплачиваемую работу, и впоследствии – начать собственное дело. А перед тем, как нырять в бассейн с акулами бизнеса, стоит узнать обо всех подводных камнях...

В далеком 1969 году некто Уильям Раш зарегистрировал патент для Television Gaming Apparatus, который использовал манипулятор для передвижения объектов на экране и их столкновения с другими объектами. Этот патент был впоследствии выкуплен компанией Magnavox, которая затем выпустила первую игровую приставку — Magnavox Odyssey, в которой использовался данный принцип. Начинающий конкурент Магнавокса, компания Atari, выпустила свою Atari 2600, куда приспособила манипуляторы, работающие по тому же принципу. Magnavox «пригрозил пальчиком» и Атари тут же приобрела лицензию на игру PONG. За последующее за этим событием деятилетие, Magnavox успешно грозил этим патентом направо и налево, пугая судом такие компании, как Seeburg, Bally-Midway, Mattel, и даже Activision и Nintendo, успешно доказывая, что патентное право сильнее любого здравого смысла.

Известно, что корпорация Immersion владеет некоторым количеством патентов на технологии Force Feedback. В частности, это U.S. Patent 6275213 и 6424333, которые описывают механизм виброотдачи на игровых контроллерах. В феврале 2002 года Immersion подала в суд на Sony и Microsoft, и затребовала с них крупный штраф за использование в Xbox и PlayStation 2 запатентованных технологий.

После выхода консоли Nintendo Wii, компания Interlink подала в суд на игрового гиганта за то, что контроллер Wiimote напоминает устройство, запатентованное ими еще в 2005 году. Впрочем, все сходство заключается в манере держать его как пульт и использовать триггер снизу – ни о каком определении положения контроллера в пространстве речи не идет.

Компания Koei, разработчик франшизы Dynasty Warriors, является обладателем совершенно абсурдного патента, который озаглавлен «Метод боя, в котором сила атаки зависит от плотности группы» (U.S. Patent 6729954).

Не так давно холдинг Paltalk подал в суд на курпнейших издателей MMO-игр. Под раздачу попали Turbine, Sony, Activision Blizzard, NCsoft и Jagex. В претензии они заявили о том, что эти компании используют принадлежащую Paltalk технологию обмена данными, которая «позволяет игрокам видеть идентичные внутриигровые площади одновременно». Этот патент был выкуплен у другой фирмы, HearMe, еще в 2002 году. В марте Paltalk уже удачно получила 90 миллонов долларов в деле против Microsoft. Тогда компания заявила, что в Halo и других многопользовательских проектах на Xbox 360 патент используется без разрешения. В результате Microsoft решила не доводить дело до длительного судебного разбирательства, предпочтя лицензировать патенты за неоглашенную сумму, что и создало неприятный для индустрии MMO-развлечений прецедент.

Радует тот факт, что сложившаяся ситуация с патентами начинает меняться в лучшую сторону. Недавно Федеральная торговая комиссия США опубликовала отчет, в котором констатируются важные проблемы патентной системы США, в частности, отрицательно влияющие на индустрию разработки программного обеспечения. В отчете также предлагаются некоторые реформы, которые потенциально могут снизить ущерб от некоторых из этих проблем.

«Пасхальные яйца» в Linux

Окно запуска Gnome

Нажимаем Alt+F2 и вводим фразу free the fish, либо gegls from outer space (подсказка: убить рыбу можно командой sudo killall gnome-panel).

Окно запуска KDE

Нажимаем Alt+F2, вводим life и получаем ответ на главный вопрос жизни.

Окно входа в систему (GDM)

Вместо логина вводим Start Dancing, для остановки вводим Stop Dancing. Также можно попробовать Require Quarter и Gimme Random Cursor.

Firefox 3

Открываем новую вкладку и вводим в адресной строке about:robots. Также стоит попробовать about:mozilla.

Google Chrome/Chromium

Нажимаем Shift+Esc, кликаем правой кнопкой мыши по любому процессу и выбираем последний пункт меню.

VLC

Просто ждем католическое рождество (25 декабря).

KDE Kexi

Создаем новую таблицу под названием Sudoku.

Amarok

Включаем композицию Майка Олдфилда «Amarok».

Google Earth

Переходим в режим исследования Марса. Вводим в окно поиска Meliza. Жмем на иконку робота.

OpenOffice.org Writer

Набираем StarWriterTeam, нажимаем F3. Также работает GoOOTeam

OpenOffice.org Calc

Просто вставьте одну из следующих функций в ячейку таблицы и нажмите Enter:

```
=GAME()
=GAME("Froggie")
=GAME(A2:C4;"TicTacToe")
=GAME("StarWars")
=ANTWORT("Das Leben, das Universum und der ganze Rest")
=STARCALCTEAM()
=TTT()
```

Vim

Набираем :help 42. Также можно попробовать следующие варианты:

```
:help holy-grail
:help!
:help map-modes
```

:help UserGettingBored
:help spoon
:help showmatch
:Ni!

Менеджеры пакетов

Просто пробуем команды из следующего списка (к примеру aptitude также можно добавить опцию -v, или -vv, или даже -vvvvv): (ubuntu) apt-get moo (ubuntu) aptitude moo (gentoo) emerge moo (gentoo) update-modules you (arch) pacman --version

ArchLinux

Добавляем в секцию [Options] файла /etc/pacman.conf строку ILoveCandy. Пробуем установить пакет.

Gentoo

В секцию FEATURES файла /etc/make.conf добавляем слово candy. Выполняем команду emerge -uDpv world.

Тайлер?

Создаем пользователя tyler, логинимся под его именем и пробуем выполнить команду halt или reboot.

Sudo

Открываем файл /etc/sudoers, находим строку Defaults и добавляем к списку опций слово insults. Набираем sudo -K, наслаждаемся результатом.

Nmap

Набираем: $ nmap -oS — scanme.nmap.org

Системный вызов reboot

```
$ grep LINUX_REBOOT_MAGIC /usr/include/linux/*.h
/usr/include/linux/reboot.h:#define LINUX_REBOOT_MAGIC1    0xfee1dead
/usr/include/linux/reboot.h:#define LINUX_REBOOT_MAGIC2    672274793
/usr/include/linux/reboot.h:#define LINUX_REBOOT_MAGIC2A   85072278
/usr/include/linux/reboot.h:#define LINUX_REBOOT_MAGIC2B   369367448
/usr/include/linux/reboot.h:#define LINUX_REBOOT_MAGIC2C   537993216
```

В последней колонке закодированы важные для Линуса (и Линукса) даты, расшифровать их можно так:
$ perl -e 'print localtime(672274793). "\n";'

Смотрим Star Wars в терминале

Это, конечно, не пасхальное яйцо, но все же: $ telnet towel.blinkenlights.nl

Это все!

Надеемся, номер вышел интересным. Если так, поддержите FPS! Отправляйте статьи, обзоры, интервью и прочее на любые темы, касающиеся игр, графики, звука, программирования и т.д. на clocktower89@mail.ru или mailto:gecko0307@gmail.com.