МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Лабораторная работа №5

«Работа с текстурной памятью»

Студент: Махлай В.С., 08-406

Преподаватель: Семенов С.А.

Москва, 2012 год

Содержание

Постановка задачи

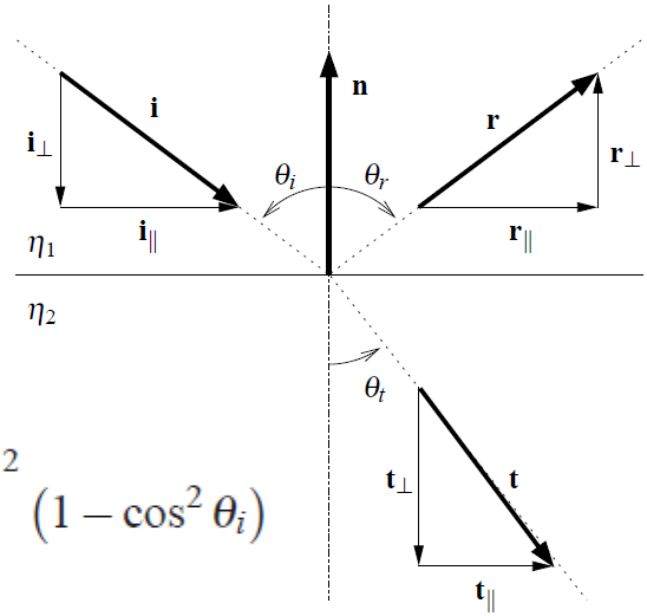
Познакомиться и освоить алгоритмы работы с задачами, предусматривающими использование текстурной памяти, этой технологии и ее реализации на языке программирования CUDA в соответствии с номером варианта задания.

Вариант №2: Трассировка лучей: Преломления.

Алгоритм

Для решения поставленной задачи были использованы материалы по данному вопросу, содержащиеся в книге А.В. Борескова, и соответствующая информация на лекции, а именно посвященная проблематике работы с текстурной памятью, преимуществами этой технологии и задачами, решающимися наиболее успешно с ее помощью.

Чтобы найти длину преломленного луча и угол, между ним и нормалью (преломленный угол), необходимо знать длину луча падающего, угол, между ним и нормалью, длину нормали и коэффициенты преломления материалов и использовать нижеприведенные формулы:



$$\sin^2\theta_t = \left(\frac{\eta_1}{\eta_2}\right)^2 \sin^2\theta_i = \left(\frac{\eta_1}{\eta_2}\right)^2 \left(1-\cos^2\theta_i\right)$$

$$\mathbf{t} = \frac{\eta_1}{\eta_2}\mathbf{i} + \left(\frac{\eta_1}{\eta_2}\cos\theta_i - \sqrt{1-\sin^2\theta_t}\right)\mathbf{n}$$

Реализация

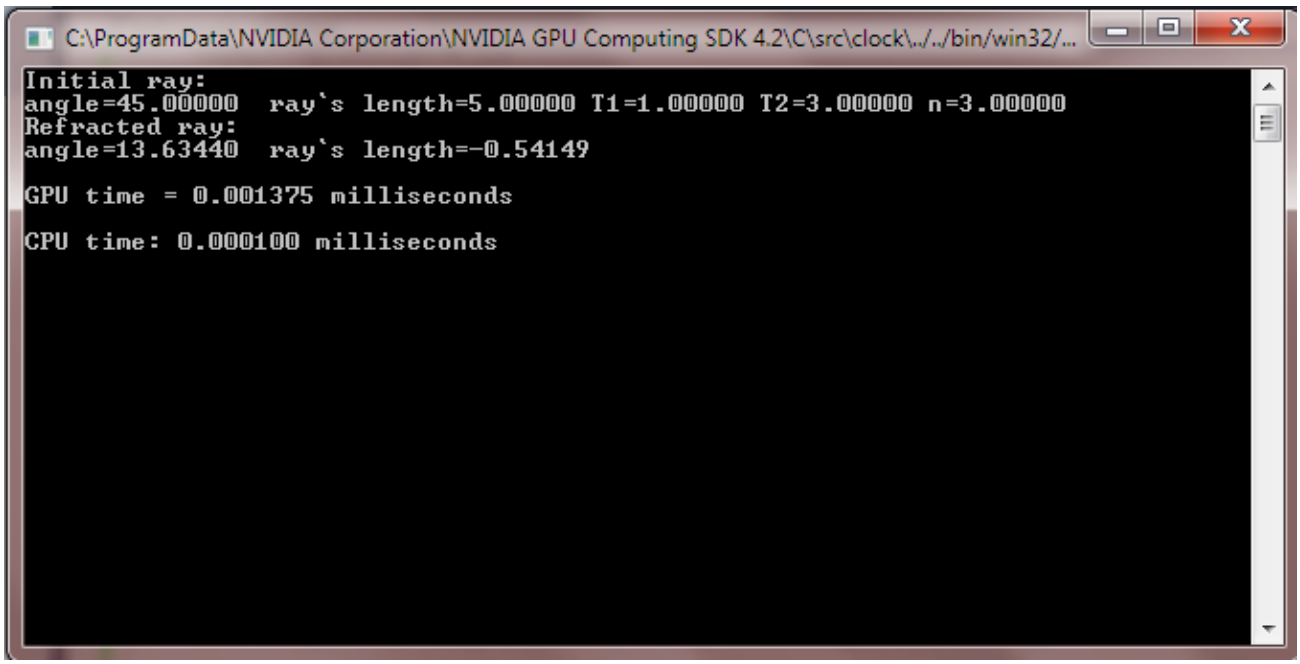
```
texture<float, 1, cudaReadModeElementType> textur;
__global__ void refract(float *res, float T1, float T2)
{
	int index = blockIdx.x * blockDim.x + threadIdx.x;
	float i = tex1Dfetch(textur,0);
	float fi_i = tex1Dfetch(textur,1);
	float n = tex1Dfetch(textur, 2);
	float fi_t = 180/3.14*asin(sqrt(sin(fi_i*3.14/180)*sin(fi_i*3.14/180))*T1/T2);
	float t = i*T1/T2+(cos(fi_i*3.14/180)*T1/T2 - sqrt(1-
sin(fi_t*3.14/180)*sin(fi_t*3.14/180)))*;
	res[0]=fi_t;
	res[1]=t;
}

int main(void)
{
	float i=5;
	float fi_i=45;
	float T1=1, T2=3;
	float n = 3;
	cudaEvent_t start;
	cudaEvent_t stop;
	float gpuTime;
	float  *ress, *resd;
	ress = (float *)malloc(3 * sizeof(float));
	cudaEventCreate(&start);
	cudaEventCreate(&stop);
	cudaMalloc((void **) &resd, 3 * sizeof(float));
	ress[0] = i; ress[1] = fi_i; ress[2] = n;
	textur.addressMode[0] = cudaAddressModeWrap;
	textur.addressMode[1] = cudaAddressModeWrap;
	textur.filterMode =  cudaFilterModePoint;
	textur.normalized = false;
	cudaMemcpy(resd, ress, 3 * sizeof(float), cudaMemcpyHostToDevice);
	cudaBindTexture(0, textur, resd, 3 * sizeof(float));
	cudaEventRecord(start, 0);
	refract <<< dim3(4096 / 32), dim3(32) >>> (resd, T1, T2);
	cudaEventRecord(stop, 0);
	cudaEventSynchronize( stop );
	cudaEventElapsedTime( &gpuTime, start, stop);
	cudaMemcpy(ress, resd, 3*sizeof(float), cudaMemcpyDeviceToHost);
	printf("Initial ray:\nangle=%.5f  ray`s length=%.5f\n", fi_i, i);
	printf("Refracted ray:\nangle=%.5f  ray`s length=%.5f\n\n", ress[0], ress[1]);
	printf("GPU time = %f milliseconds\n\n", gpuTime / 100);

	clock_t cpu_start = clock();
	refract(ress, T1, T2, i, fi_i, n);
	clock_t cpu_finish = clock();
	printf("CPU time: %f milliseconds\n", (float)(cpu_finish - cpu_start)/
(float)CLOCKS_PER_SEC);
	free(ress);
	cudaFree(resd);
}
```

Результаты

В результате выполнения данной программы мы имеем возможность моделирования процесса преломления лучей в процессе трассировки лучей, что является хорошим примером программы, реализуемой с помощью технологии работы с текстурной памятью.

Сравнение результатов производительности CPU/GPU

Характеристики ПК: 32-х разрядная операционная система Windows 7 Максимальная, процессор: Intel Core2 Quad CPU Q6600 2.40 GHz, 4 ГБ оперативной памяти, видеокарта NVIDIA GeForce 9800 GTX/9800 GTX+.



Выводы

В ходе выполнения данной лабораторной работы были получены знания по работе с текстурной памятью на примере реализации поставленных задач с использованием языка программирования CUDA, что предоставляет нам новые возможности при работе на GPU.

Список литературы

1.Боресков А.В., Харламов А.В. «Основы работы с технологией CUDA». Издательство ДМК Пресс, 2010, 232 стр.

2.Казённов А.М. Основы технологии CUDA // Компьютерные исследования и моделирование 2010 Т. 2 № 3 С. 295–308

3.Материалы из Интернета.

Список сокращений

CPU – центральный процессор,

GPU – графический процессор,

CUDA – Compute Unified Device Architecture.