ASSIGNMENT - III

```
// mandelbrot_pbo_cpu_gpu.cu
#include <windows.h>
#include <GL/gl.h>
#include <cuda_runtime.h>
#include <cuda_gl_interop.h>
#include <iostream>
#include <chrono>
#include <cmath>

#define WIDTH 1024
#define HEIGHT 1024
#define MAX_IT 1000

// ---------------- CPU ----------------
void mandelbrotCPU(unsigned char* img, int width, int height, double cx, double cy, double scale){
    for(int py=0; py<height; py++){
        for(int px=0; px<width; px++){
            double x0 = (px - width/2.0) * scale + cx;
            double y0 = (py - height/2.0) * scale + cy;
            double x=0,y=0;
            int iter=0;
            while(x*x+y*y<=4 && iter<MAX_IT){
                double xt = x*x - y*y + x0;
                y = 2*x*y + y0;
                x = xt;
                iter++;
            }
            int idx = 3*(py*width + px);
            unsigned char color = (unsigned char)(255*iter/MAX_IT);
            img[idx+0] = color;
            img[idx+1] = color;
            img[idx+2] = color;
```

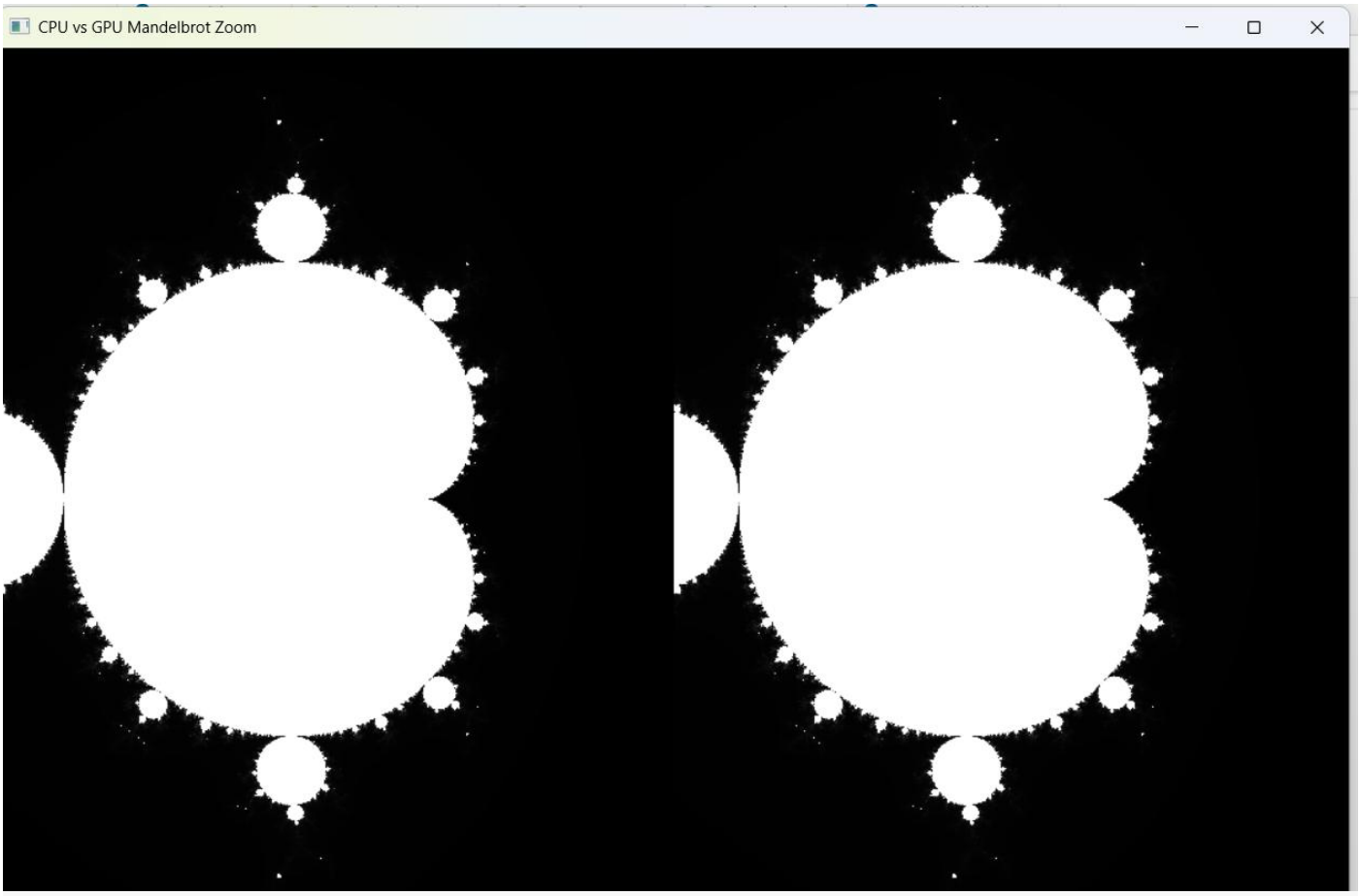
```
        }
    }
}

// ---------------- GPU ----------------
__global__ void mandelbrotGPU(unsigned char* img, int width, int height, double cx, double cy, double scale){
    int px = blockIdx.x*blockDim.x + threadIdx.x;
    int py = blockIdx.y*blockDim.y + threadIdx.y;
    if(px>=width || py>=height) return;

    double x0 = (px - width/2.0) * scale + cx;
    double y0 = (py - height/2.0) * scale + cy;
    double x=0,y=0;
    int iter=0;
    while(x*x+y*y<=4 && iter<MAX_IT){
        double xt = x*x - y*y + x0;
        y = 2*x*y + y0;
        x = xt;
        iter++;
    }
    int idx = 3*(py*width + px);
    unsigned char color = (unsigned char)(255*iter/MAX_IT);
    img[idx+0] = color;
    img[idx+1] = color;
    img[idx+2] = color;
}

// ---------------- Display ----------------
void display(unsigned char* cpuImg, GLuint pbo){
    glClear(GL_COLOR_BUFFER_BIT);

    // CPU left half
    glRasterPos2i(-1,-1);
```

```
    glDrawPixels(WIDTH/2, HEIGHT, GL_RGB, GL_UNSIGNED_BYTE, cpuImg);

    // GPU right half (draw PBO as pixel data)
    glBindBuffer(GL_PIXEL_UNPACK_BUFFER, pbo);
    glRasterPos2i(0,-1);
    glDrawPixels(WIDTH/2, HEIGHT, GL_RGB, GL_UNSIGNED_BYTE, 0);
    glBindBuffer(GL_PIXEL_UNPACK_BUFFER, 0);

    glFlush();
}

// ---------------- Win32 OpenGL ----------------
HWND createWindow(HINSTANCE hInstance, int width, int height){
    WNDCLASS wc={0};
    wc.style = CS_OWNDC;
    wc.lpfnWndProc = DefWindowProc;
    wc.hInstance = hInstance;
    wc.lpszClassName = "MandelbrotPBOCPU_GPU";
    RegisterClass(&wc);
    return CreateWindowExA(0, wc.lpszClassName, "CPU vs GPU Mandelbrot PBO",
        WS_OVERLAPPEDWINDOW|WS_VISIBLE, 100,100,width,height,nullptr,nullptr,hInstance,nullptr);
}

// ---------------- Main ----------------
int main(){
    HINSTANCE hInstance = GetModuleHandle(nullptr);
    HWND hwnd = createWindow(hInstance, WIDTH, HEIGHT);
    HDC hdc = GetDC(hwnd);

    PIXELFORMATDESCRIPTOR pfd={sizeof(PIXELFORMATDESCRIPTOR),1};
    pfd.dwFlags = PFD_DRAW_TO_WINDOW|PFD_SUPPORT_OPENGL|PFD_DOUBLEBUFFER;
    pfd.iPixelType = PFD_TYPE_RGBA;
    pfd.cColorBits = 24;
    int pf = ChoosePixelFormat(hdc,&pfd);
```

```
SetPixelFormat(hdc,pf,&pfd);
HGLRC glrc = wglCreateContext(hdc);
wglMakeCurrent(hdc,glrc);

// ---------------- Setup ----------------
unsigned char* cpuImg = new unsigned char[3*WIDTH*HEIGHT/2];
GLuint pbo;
glGenBuffers(1,&pbo);
glBindBuffer(GL_PIXEL_UNPACK_BUFFER,pbo);
glBufferData(GL_PIXEL_UNPACK_BUFFER,3*WIDTH*HEIGHT/2,0,GL_DYNAMIC_DRAW);
glBindBuffer(GL_PIXEL_UNPACK_BUFFER,0);
cudaGraphicsResource* cudaPBO;
cudaGraphicsGLRegisterBuffer(&cudaPBO, pbo, cudaGraphicsMapFlagsWriteDiscard);
double cx=0, cy=0, scale=4.0/WIDTH;
MSG msg;
int frame=0;
bool running=true;
while(running){
   while(PeekMessage(&msg,nullptr,0,0,PM_REMOVE)){
      if(msg.message==WM_QUIT) running=false;
      TranslateMessage(&msg);
      DispatchMessage(&msg);
   }

   // Zoom in
   scale *= 0.99;
   cx += 0.001*frame;
   cy += 0.001*frame;

   // CPU timing
   auto startCPU = std::chrono::high_resolution_clock::now();
   mandelbrotCPU(cpuImg, WIDTH/2, HEIGHT, cx, cy, scale);
   auto endCPU = std::chrono::high_resolution_clock::now();
   double cpuTime = std::chrono::duration<double,std::milli>(endCPU-startCPU).count();
```

```
        // GPU timing (direct PBO)
        cudaEvent_t start, stop;
        cudaEventCreate(&start);
        cudaEventCreate(&stop);
        cudaEventRecord(start);
        unsigned char* d_ptr;
        size_t num_bytes;
        cudaGraphicsMapResources(1,&cudaPBO,0);
        cudaGraphicsResourceGetMappedPointer((void**)&d_ptr,&num_bytes,cudaPBO);
        dim3 threads(16,16);
        dim3 blocks((WIDTH/2+15)/16,(HEIGHT+15)/16);
        mandelbrotGPU<<<blocks,threads>>>(d_ptr, WIDTH/2, HEIGHT, cx, cy, scale);
        cudaGraphicsUnmapResources(1,&cudaPBO,0);
        cudaDeviceSynchronize();
        cudaEventRecord(stop);
        cudaEventSynchronize(stop);
        cudaEventElapsedTime(&gpuTime,start,stop);
        cudaEventDestroy(start);
        cudaEventDestroy(stop);
        display(cpuImg,pbo);
        SwapBuffers(hdc);
        std::cout << "Frame " << frame << ": CPU = " << cpuTime << " ms, GPU = " << gpuTime << " ms\n";
        frame++;
    }
    delete[] cpuImg;
    glDeleteBuffers(1,&pbo);
    cudaGraphicsUnregisterResource(cudaPBO);
    wglDeleteContext(glrc);
    ReleaseDC(hwnd,hdc);
    DestroyWindow(hwnd);
    cudaDeviceReset();
    return 0;
}
```

OUTPUT




```
(base) PS C:\Users\Karunya\Documents\Sem 7 - LAs\GPA\Assignments> .\cuda_gl.exe
Frame 0: CPU = 607.265 ms, GPU = 24.0005 ms
Frame 1: CPU = 611.409 ms, GPU = 22.6244 ms
Frame 2: CPU = 618.315 ms, GPU = 23.0625 ms
Frame 3: CPU = 635.971 ms, GPU = 22.785 ms
Frame 4: CPU = 642.833 ms, GPU = 23.9929 ms
Frame 5: CPU = 656.31 ms, GPU = 23.7465 ms
Frame 6: CPU = 661.641 ms, GPU = 26.5343 ms
Frame 7: CPU = 674.409 ms, GPU = 24.7143 ms
Frame 8: CPU = 682.057 ms, GPU = 25.1862 ms
Frame 9: CPU = 690.687 ms, GPU = 28.2212 ms
Frame 10: CPU = 704.778 ms, GPU = 67.7139 ms
Frame 11: CPU = 712.698 ms, GPU = 43.6311 ms
Frame 12: CPU = 723.084 ms, GPU = 38.8016 ms
Frame 13: CPU = 727.906 ms, GPU = 78.4731 ms
Frame 14: CPU = 738.505 ms, GPU = 27.8166 ms
Frame 15: CPU = 755.246 ms, GPU = 26.7133 ms
Frame 16: CPU = 771.099 ms, GPU = 26.7386 ms
Frame 17: CPU = 766.832 ms, GPU = 45.0931 ms
Frame 18: CPU = 773.88 ms, GPU = 128.001 ms
Frame 19: CPU = 773.968 ms, GPU = 89.8072 ms
(base) PS C:\Users\Karunya\Documents\Sem 7 - LAs\GPA\Assignments>
```

Figure 1: Graphics Program Using OpenGL and CUDA.