TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

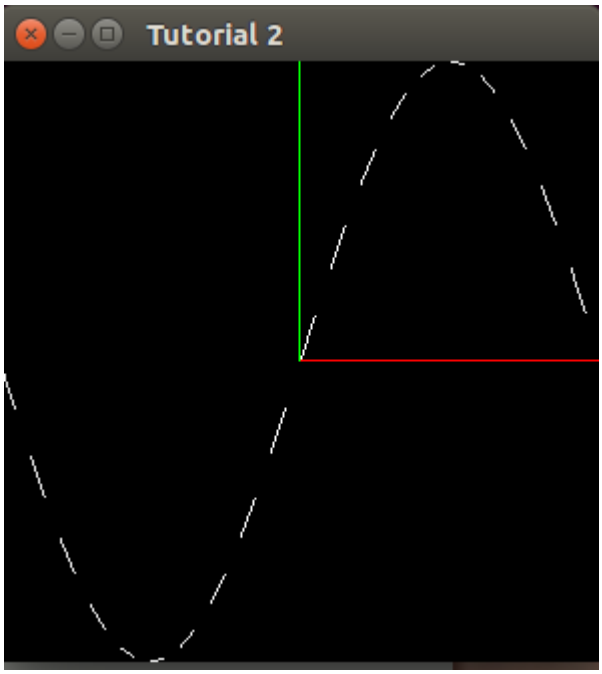
강사 : Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 : 황수정
sue100012@naver.com
71 일차 (2018. 06. 11)

sudo apt-get update
sudo apt-get install build-essential
sudo apt-get install freeglut3 freeglut3-dev
sudo apt-get install glew-utils glee-dev
sudo apt-get install libglew-dev

→ 업데이트 후에
sin(wt)를 이산신호로 만들어보라

코드를 만들고 나서
gcc non_anim_sin.c -lGL -lglut -lGLU -lm

와 같은 방식으로 옵션을 줘서 컴파일 하면 된다.



와 같이 뜬다.

```
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <GL/glut.h>
#include <GL/glu.h>
#include <GL/gl.h>
#include <GL/freeglut.h>

void originAxis(void);
void sineWave(void);
void idle(void);

void display(void)
{
	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
	//이전 설정들 다 날려버리라는 뜻
```

```
	originAxis();  //좌표 축 설정
	sineWave();

	//printf("%s\n", gluErrorString(glGetError()));
	glutSwapBuffers();
}

void sineWave(void)		//v=fx c=fx
{
	float wavelength = 2;		//주기
	float amplitude = 1;		//진폭
	float inc = 0.05;		//증가폭(샘플링 주기)
	float k, x, y;			//각주파수
	glBegin(GL_LINES);		//그림을 그리겠다.
	glColor3f(1,1,1);		//RGB 색으로 흰색으로 색이 나옴
	for(x=-1;x<=1;x+=inc){
		k = 2 * 3.14 / wavelength;	//
		y = amplitude * sin(k * x);	//진폭을 크게 주면 크게 진동하고 작게 주면 작아짐
		glVertex3f(x, y, 0);	// x= 0, 1, 2, ...n 일 때,y =  0, 1, 2, ...n 으로 그려진다.
	}
	glEnd();

}//어떻게 설정해주냐에 따라 sin 인지 cos 인지를 그릴 수 있다.

void idle(void)		// 아무것도 건들이지 않았을 때, 동작하는 함수
{
	float wavelength = 2;
	float amplitude = 1;
	float inc = 0.05;
	float k, x, y;
	for (x = -1; x <= 1; x += inc){
		glBegin(GL_POINTS);
		glPointSize(200);
		glColor3f(0, 1, 0);
		k = 2 * 3.14 / wavelength;
		y = amplitude * sin(k * x);
		glVertex3f(x, y, 0);
		glEnd();
	}

	glutPostRedisplay();
}sudo apt-get install libglew-dev

void originAxis(void)
{
	glBegin(GL_LINES);		//그리는 것을 시작하는 것 end 까지 그려라
	glColor3f(1,0,0);		//RGB 를 기준으로 하는 것 빨간색 선을 그리겠다
	glVertex3f(0,0,0);		// 위의 빨간색 선을 원점 부터
	glVertex3f(1, 0, 0);		// 1, 0, 0 까지 그리겠다.
	GlColor3f(0,1,0);		//초록색을 그리겠다.
```

```
	glVertex3f(0,0,0);
	glVertex3f(0, 1, 0);
	glColor3f(0,0,1);		//파란색을 그리겠다.
	glVertex3f(0,0,0);
	glVertex3f(0, 0, 1);
	glEnd();
}//z 축은 우리가 2d 이기 때문에 그려지지 않는다.

int main(int argc, char **argv)
{
	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE | GLUT_DEPTH);
	glutCreateWindow("Tutorial 2");

	glOrtho(-1, 1, -1, 1, -1, 1);
	glEnable(GL_DEPTH_TEST);

	glutDisplayFunc(display);	//화면에 출력하라는 것 //display 가 그리는 함수
	glutIdleFunc(idle);
	glutMainLoop();			//실제로 화면에 그리기 시작

	return EXIT_SUCCESS;
}

#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>

#include <GL/glut.h>

#define SLICE		360

void draw_omega_sin(void);

float common_angles[5] = {15.0, 30.0, 45.0, 60.0, 75.0};
float freq_table[5] = {1000.0, 2400.0, 5000.0, 24000.0, 77000.0};

float theta = 0.0;

void display(void)
{
	glClearColor(0.0, 0.0, 0.0, 1.0);
	//벡터가 하나 추가되었다. 뒤에는 알파값으로 투명도를 뜻함
	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
	//버퍼를 비우는 작업을 해주는 것이다. 잔상이 보이지 않게 해주려고 하는 것이다.
	//이전 내용을 지워주는 것이다.(버퍼 지워지지 않고 현 동작하고 같이 보이면
잔상으로 보임)
	glLoadIdentity();

	//gluLookAt(0.0, 0.0, 3.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);
```

```
	glColor3f(1, 0, 0);
	//RGB 칼라를 말하는 것

	glBegin(GL_LINE_LOOP);
	glVertex3f(100.0, 0.0, 0.0);
	glVertex3f(-100.0, 0.0, 0.0);
	glEnd();

	glColor3f(0.0, 1.0, 0.0);

	glBegin(GL_LINE_LOOP);
	glVertex3f(0.0, 100.0, 0.0);
	glVertex3f(0.0, -100.0, 0.0);
	glEnd();

	draw_omega_sin();
	glutSwapBuffers();
}

#if 0
void reshape(int w, int h)
{
	glViewport(0, 0, (GLsizei)w, (GLsizei)h);
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	gluPerspective(60, (GLfloat)w / (GLfloat)h, 0.1, 100.0);
	glMatrixMode(GL_MODELVIEW);
}
#endif

void reshape(int w, int h)
{
	GLfloat n_range = 100.0f;

	if(h == 0)
		h = 1;

	glViewport(0, 0, w, h);
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();

	if(w <= h)
		glOrtho(-n_range, n_range, -n_range * h / w, n_range * h / w, -n_range, n_range);
	else
		glOrtho(-n_range * w / h, n_range * w / h, -n_range, n_range, -n_range, n_range);

	glMatrixMode(GL_MODELVIEW);
	glLoadIdentity();
}

void keyboard(unsigned char key, int x, int y)
```

```
{
	switch(key)
	{
		case 27:
			exit(0);
			break;
	}
}

void set_rand_amplitude(float *amp)
{
	*amp = rand() % 3 + 3;
}

void set_angle_with_common_angles(float *angle)
{
	*angle = common_angles[rand() % 5];
}

void angle2radian(float *angle, float *radian)
{
	*radian = *angle * M_PI / 180.0;
}

void radian2angle(float *angle, float *radian)
{
	*angle = *radian * 180.0 / M_PI;
}

void set_rand_frequency(float *freq)
{
	*freq = freq_table[rand() % 5];
}

void calc_period(float *freq, float *period)
{
	*period = 1 / (*freq);
}

void calc_angular_velocity(float *freq, float *ang_vel)
{
	*ang_vel = 2 * M_PI * (*freq);
}

float get_step(float slice, float period)
{
	return period / slice;
}

void cos_sim(float amplitude, float ang_vel, float period)
{
	int cnt = 0;
```

```
	float step, t = 0.0;

	t = step = get_step(SLICE, period);

	while(cnt++ < 36)
	{
		printf("%.1fcos(%f * %.8f) = %f\n", amplitude, ang_vel,
			t, amplitude * cos(ang_vel * t));
		t += step;
	}
}

void sin_sim(float amplitude, float ang_vel, float period)
{
	int cnt = 0;
	float step, t = 0.0;

	t = step = get_step(SLICE, period);

	while(cnt++ < 36)
	{
		printf("%.1fsin(%f * %.8f) = %f\n", amplitude, ang_vel,
			t, amplitude * sin(ang_vel * t));
		t += step;
	}
}

void draw_omega_sin(void)
{
	float amp, angle, period, freq, rad, omega, t, step = 0.0;
	float radius = 3.0;
	float x = 0, x2 = 0, y2, cx, cy;
	float tmp;
	int cache = 0;

	srand(time(NULL));

#if 0
	set_rand_amplitude(&amp);
	set_angle_with_common_angles(&angle);
	angle2radian(&angle, &rad);
	set_rand_frequency(&freq);
	calc_period(&freq, &period);
	calc_angular_velocity(&freq, &omega);
#endif

#if 1
	amp = 10;
	angle = 45.0;
	freq = 100.0;

	angle2radian(&angle, &rad);		//
```

```
	calc_period(&freq, &period);		//주파수 정보가 있으니까 주기를 알 수 있어서(역수)
	calc_angular_velocity(&freq, &omega);	//각속도 = 신호처리에서는 각주파수
#endif

#if 0
	printf("amplitude = %f\n", amp);
	printf("angle = %f degree\n", angle);
	printf("radian = %f\n", rad);
	printf("frequency = %f\n", freq);
	printf("period = %f\n", period);
	printf("angular_velocity = %f\n", omega);
#endif

	t = step = get_step(SLICE, period);
	//step 각 단위별로 등분하자. 등분 될 수록 정보가 세밀해질 것이다.
	//printf("t = %f\n", t);
#if 1
	if(t > period)
		t = 0.0;
#endif

	glBegin(GL_LINES);
	for(; ; t += step)
	{
		if(t > 3 * period)	//3 주기보다 크다는 3 번 진동하고 끝난다는 뜻이다.
		{
			break;
			t = 0.0;
		}

		//float rad_angle = angle * (M_PI / 180.0);
		//x2 += x;			// time += step;
		//x2 += 0.1;
		y2 = amp * sin(omega * t);
		//y2 = radius * sin((double)rad_angle);

		if(cache)
		{
			glVertex2f(cx * 4000, cy);
			glVertex2f(t * 4000, y2);
		}

		cache = 1;
		cx = t;
		cy = y2;
		//printf("t = %f, y2 = %f\n", t * 4000, y2);
	}
	glEnd();
}

int main(int argc, char **argv)
{
```

```
	float amplitude, angle, period, frequency, radian, angular_velocity;
	float step = 0.0;

	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_DOUBLE);
	glutInitWindowSize(1200, 800);
	glutInitWindowPosition(0, 0);
	glutCreateWindow("Digital Signal Processing");

#if 0
	srand(time(NULL));

	set_rand_amplitude(&amplitude);
	set_angle_with_common_angles(&angle);
	angle2radian(&angle, &radian);
	set_rand_frequency(&frequency);
	calc_period(&frequency, &period);
	calc_angular_velocity(&frequency, &angular_velocity);

	printf("amplitude = %f\n", amplitude);
	printf("angle = %f degree\n", angle);
	printf("radian = %f\n", radian);
	printf("frequency = %f\n", frequency);
	printf("period = %f\n", period);
	printf("angular_velocity = %f\n", angular_velocity);

	cos_sim(amplitude, angular_velocity, period);
	sin_sim(amplitude, angular_velocity, period);
#endif

	glutDisplayFunc(display);
	//glutIdleFunc(display);
	glutReshapeFunc(reshape);
	//glutKeyboardFunc(keyboard);
	glutMainLoop();

	return 0;
}
```