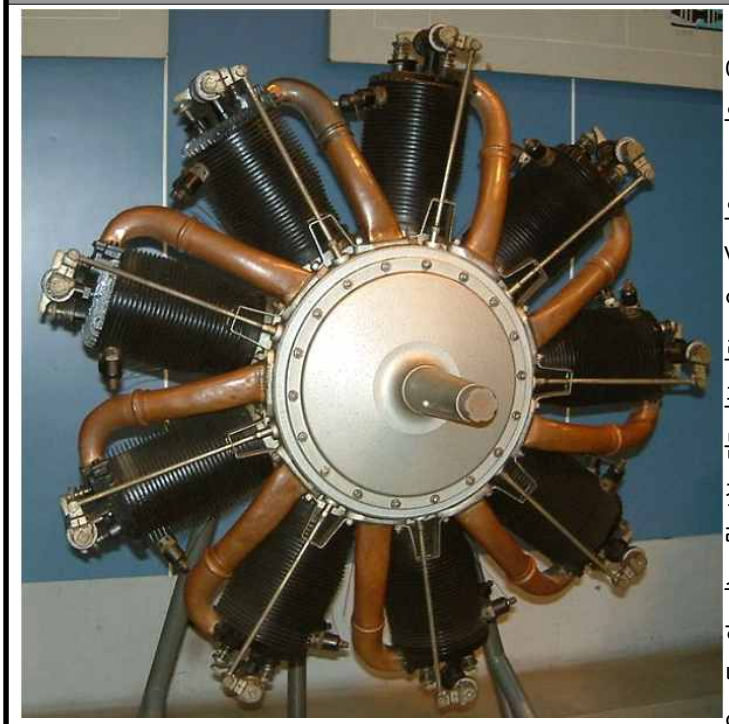
Project Proposal

과목	컴퓨터 응용 설계	성명	Amylose
제목	내연기관 엔진 시뮬레이터 (부제 : Radial Engine 시뮬레이터)		
목표	기구학적 지식과 openGL의 애니메이션 기법을 활용하여 Radial Engine을 모사한다.		

분야에 대한 배경



실린더가 크랭크축을 중심으로 원형으로 배치되어 마치 별같이 보이기 때문에 성형엔진으로 불리우는 왕복엔진이다. 발명자는 조지 웨스팅하우스.

이 성형엔진의 큰 장점은 별도의 냉각장치가 필요 없다는 점이다. 기존에 사용되던 선형엔진이나 V형엔진의 경우 프로펠러로의 공랭이 실린더의 배열 때문에 거의 불가능하지만, 이 엔진의 경우 프로펠러 바로 뒤에 실린더가 원형으로 배치되므로 프로펠러 바람으로 바로바로 방열이 가능하기 때문이다. 이로인한 이점은 크게 두가지가 있는데, 첫 번째로 피탄에 취약한 수냉식 엔진 대신에 공랭식 구조라 내구도가 높아진다는 점, 두 번째로 수냉장치가 생략되므로, 기존엔진과 비교하여 상당히 가볍게 만들 수 있다는 점 이다. 이 중량문제는 냉각장치 말고도 크랭크 축 자체를 짧게 만들수도 있기 때문에 기존보다 더욱 가볍게 만드는 것도 가능했다.

당연하게도, 단점또한 있는데 가장 대표적인 것은 엔진의 면적이다. 별모양으로 실린더가 붙어있기 때문에 자연스럽게 정면에서의 단면적이 넓어질 수밖에 없고, 이는 항공기 성능의 지표중 하나인 항력(공기저항)에 영향을 끼치게 된다.

제 2차 세계대전때만 해도 높은 내구도와 공랭식 엔진의 내구성 덕분에 현역으로 쓰였다. 실제로 성형엔진을 장착한 비행기는 공격을 받고 실린더 몇 개가 망가졌음에도 불구하고, 무사히 착륙했다. 그러나 가스터빈엔진이 나오고 난뒤로부터는 퇴조하였다. 가볍고 간단하다는 구조 덕분에 아직 경비행기에는 많이 쓰인다.

분야에 대한 공학적 배경 및 공식

Design a Radial Engine

1. Radial Engine

The Radial Engine is a reciprocating type internal combustion engine configuration in which the cylinders point outward from a central crankshaft like the spokes on a wheel.

This type of engine was commonly used in most of the aircrafts and tanks.

In a radial engine, the pistons are connected to the crankshaft with a master-and-articulating-rod assembly.

Four stroke radials always have an odd number of cylinders per row, so that a consistent every-other-piston firing order can be maintained, providing smooth operation.

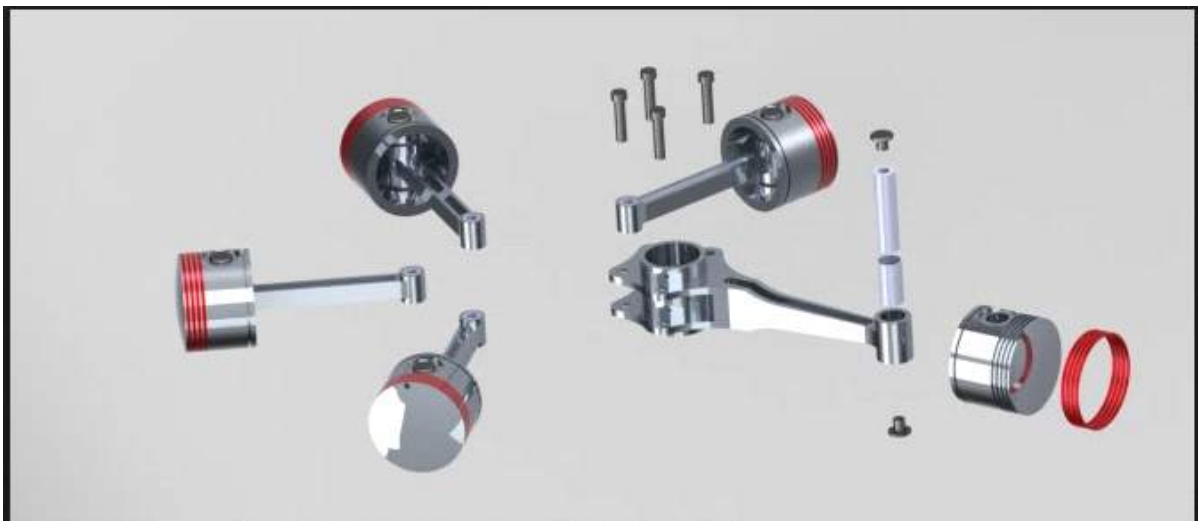
The four stroke consequence of every engine is:

a). intake

b). Compression

c). Power

d). Exhaust



2. Design

Radial Engine Characteristics

Rpm=6000

Piston diameter Dp=70mm

Master-rod length Lmr=120mm

Crank length Rcr=30mm

(1) Kinematical and Dynamical Calculations

(a). Ratio

Between the crank of the crankshaft and the master-rod length:

$$\lambda = \frac{R_{cr}}{L_{mr}} = \frac{0.03}{0.12} = 0.25$$

(b). Angular velocity

Specifies the angular velocity of the object and the axes about which the object is rotating:

$$\omega = \frac{\pi \cdot n}{30} = \frac{3.14 * 6000}{30} = 328$$

(c) Current Piston Stroke

Reciprocating motion, used in reciprocating engines and other mechanisms is back-and forth motion. Each cycle of reciprocation consists of two opposite motions, there is a motion in one direction and then a motion back in the opposite direction. Each of them is called a stroke.

$$S_h = R[(1 - cos\emptyset) + \frac{\lambda}{4}(1 - cos2\phi)]$$

(d) Area of the piston head:

$$F_p = \frac{\pi D^2}{4} = \frac{3.14 * (70.1^{-3})^2}{4} = 38465.1^{-6}$$

So Fp=0.00384m2

(e) Different forces acting on the master-rod:

Gas Force, Pg, N-Analytical calculation of the gas forcesas a function of the angle of rotation of the crankshaft ϕ

Is done according to the next formula:

$$P_g = [p_b \left(\frac{S_h + S_c}{S_c + S_x}\right)^2 - p_{opp.}]F_p$$

$$B = \left(\frac{S_h + S_c}{S_c + S_x}\right)^2$$

Where, Sh: Working stroke;

Sc: The stroke according to the height of the combustion chamber

Popp.= 0.1Mpa: The pressure acting on the opposite side of the piston

It is equal to this in 4-stroke engines

Pb: That is the pressure in the beginning

$$\phi = \frac{0}{180^o} \longrightarrow p_b = p_a = 0.13MPa$$

$$\phi = \frac{0}{360^o} \longrightarrow p_b = p_a = 0.13MPa$$

$$\phi = \frac{360^o}{540^o} \longrightarrow p_b = p_j = 0.611MPa$$

$$\phi = \frac{540^o}{720^o} \longrightarrow p_b = p_r = 0.126MPa$$

n-indicator that is changing in the following borders:

$$\phi = \frac{0}{180^o} \longrightarrow n = 0;$$

$$\phi = \frac{180^o}{360^o} \longrightarrow n = n_1 = 1.375;$$

$$\phi = \frac{360^o}{540^o} \longrightarrow n = n_2 = 1.25;$$

$$\phi = \frac{540^o}{720^o} \longrightarrow n = 0;$$

Inertia Force

(i) Inertia Forces of the objects with linear motions:

$$P_j = -m_j\omega^2 R[\frac{\cos(\phi+\beta)}{cos\beta} + \lambda\frac{cos^2(\phi)}{cos^2\beta}]10^{-6}(MN)$$

(ii) Inertia Force of the objects with radial motion:

$$P_R = 10^{-6}m_R\omega^2 R = const(MN)$$

*the masses mj and mr are defined as following:

$$m_j = m_{p\,gr} + m_{mr\,gr}$$

$$m_R = m_{cr} + m'_{mr\,gr}$$

Where

mcr: is the mass of the crank

$m'_{mr\,gr}$: is the mass of the part of master-rod that is brought to the axis of the crank

$m_{p\,gr}$: is the mass of the piston group

$m_{mr\,gr}$: is the mass of the part of the master-rod that is brought to the axis of the piston bolt

(iii) Other Forces and moments acting on the crankshaft mechanism, resulting of the forces of the gas, the inertia forceand centrifugal forces the crankshaft mechanism is loaded with forces that could be calculated analytically

$N = P_{\Sigma}.\tan\beta$ - outside forces, MN

$S = P_{\Sigma}.\frac{1}{\cos\beta}$ - the force acting on the axis of the master-rod

$T = P_{\Sigma}.\frac{\sin(\varphi+\beta)}{\cos\beta}$ - tangential force

$Z = P_{\Sigma}.\frac{\cos(\varphi+\beta)}{\cos\beta}$ - normal force

Where:

$P_{\Sigma} = P_g + P_j$

(iv). Calculation of the outer diameter of the flywheel, it is determined by the following formula:

$$D_m = \frac{60}{\pi.n}.V_{per.} = \frac{60}{3,14.6000}.75 = 0,239m$$

Where: Vper: the peripheral velocity which is in the following boundaries

Vper= (50/100)m/s

프로젝트 계획	
1단계	3D 모델링 프로그램에서 저장된 파일을 *.STL혹은 *.OBJ파일로 변환한 후 openGL 프로그램으로 불러들이는 것을 성공시킨다. openGL 코드상으로 모델링하는 것은 기술적/시간적으로 효율적이지 못하고 시각적으로 깔끔하고 아름답지 못하기 때문에 3D모델링 프로그램으로(혹은 사용할 수 있다면 3D디자인 프로그램으로) 모델링을 한 후에 이것을 불러들여 시각적으로 보여지는 것을 최우선적으로 성공시켜야 한다.
2단계	1개의 실린더 와 피스톤을 움직인다. 즉 왕복운동을 회전운동으로 바꾸는 애니메이션을 실시한다. 기구학적 공식 및 기타 필요한 공학적인 수식등을 활용하여 엔진의 움직임을 모사한다.
3단계	1개의 실린더 및 피스톤을 scanf나 GUI의 버튼등을 통해서 제어한다. 이를테면 회전수를 입

	력하여 회전속도등을 제어하여 눈으로 확인할 수 있도록 한다.
4단계	성형엔진을 모사하기 위하여, 여러개의 실린더 및 피스톤을 움직인다. 각 실린더 및 피스톤의 왕복운동을 회전운동으로 변환하기 위해서 실제 성형엔진의 실린더별 점화 타이밍을 알아본다. 또한 기구학적 공식 및 기타 필요한 수학적 수식등을 알아본 점화 타이밍과 조합하여 엔진의 움직임을 모사한다.
5단계	성형엔진을 scanf나 GUI의 버튼등을 통해서 제어한다. 이를테면 회전수를 입력하여 회전속도등을 제어하여 눈으로 확인할 수 있도록 한다.
총평	총 5단계의 스테이지를 현재 남아있는 학기의 시간별로 잘 배분하여 프로젝트를 실시한다. 1단계 : 가장 우선적으로 성공 및 완성되어야 할 작업이므로 1주~2주정도 시간을 소모하여 확실히 알아두고 응용할 수 있도록 한다. 2단계 : openGL의 애니메이션 구현이 프로젝트의 Critical Point이므로 약 2주정도 시간을 소모하여 기본적인 예제부터 응용방법까지 알아보고 적용한다. 3단계 : 애니메이션을 제어하는 방법이므로 1주 이내에 완성한다. 4단계 : 2단계의 응용이므로, 1주 이내에 완성하여 성공 및 적용시킨다. 5단계 : 이 프로젝트의 최종 목표이므로 1주~2주 이내에 완성시킨다.

프로젝트 타임 테이블

STAGE	7주차	8주차	9주차	10주차	11주차	12주차	13주차	14주차	15주차
1단계									
2단계									
3단계									
4단계									
5단계									
발표									

기반 시스템 및 프로그램

● **Windows Based 1**

- OS : Windows 10 Pro X64
- H/W : AMD A8-6410 APU, 8GB RAM, AMD Radeon R5 & AMD Radeon HD8500 Dual GPU
- Compiler : MinGW-w64 - GCC - 6.1.0 X64
- IDE S/W : Code::Blocks, Gedit Editor
- openGL Lib : freeglut latest ver, GLEW also.

● **Windows Based 2**

- OS : Windows 10 Pro X64
- H/W : Intel i5 2500 CPU, 8GB RAM, Nvidia GT210 GPU
- Compiler : VC++ - 2010 X64
- IDE S/W : Code::Blocks, Pluma Editor, QT5
- openGL Lib : Freeglut latest ver, GLEW also.

● **Linux Based**

- OS : HamoniKR R2.1(LinuxMint 17.3 Rosa) X64
- H/W : Intel C2D E8400 CPU, 8GB RAM, Nvidia 9600GT GPU
- Compiler : GNU - GCC - 6.2.0 X64
- IDE S/W : Code::Blocks, Pluma Editor, QT5
- openGL Lib : Freeglut latest ver, GLEW also.

필요한 공식, 능력, 기술

〈변환행렬식〉

1) Translation Matrix

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$: [x,y,z]점을 각각 t의 해당성분 요소많큼 이동시키는 Matrix

2) Rotation Matrix

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$: x좌표에 대하여 Θ많큼 회전시키는 Matirx

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$: y좌표에 대하여 Θ많큼 회전시키는 Matirx

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$: z좌표에 대하여 Θ많큼 회전시키는 Matirx

3) Scaling

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$: S량많큼 크기를 변환하는 Matrix

4) 4x4 Homogeneous Transformation Matirx

$P^0 = T_1^0 P^1$일 때, $T_1^0 = \begin{bmatrix} R_1^0 & d_1^0 \\ 0 \ 0 \ 0 & 1 \end{bmatrix}$

R_1^0 : {1} 좌표계의 단위백터들을 {0} 좌표계에 투영시켜 구함

d_1^0 : {0} 좌표계에 대한 {1} 좌표계의 원점좌표

5) Viewing Coordinate System (VCS) -> 물체상에 원점이 존재하며 화면좌표계 평행한 좌표계
Projection이 일어나기 전에 VCS으로 모든 좌표값들이 변환되어야 한다.

- view site : 시선이 향하는 물체의 한점
- view point : 눈의 위치

$$\overrightarrow{view\,v} = \frac{viewsite - viewpoint}{|viewsite - viewpoint|}$$

- up vector : SCS의 y축의 방향백터와 관계
- image plane

$$T_v^w = \begin{bmatrix} R_v^w & p_v^w \\ 0\ 0\ 0 & 1 \end{bmatrix} = \begin{bmatrix} x_v^w\ y_v^w\ z_v^w\ p_v^w \\ 0\ 0\ 0\ 1 \end{bmatrix}$$

z_v^w = -view vector

y_v^w = up vector

x_v^w = y_v^w x z_v^w

p_v^w = view site

6) Projection : 시야 이를테면, 투상도

->Perspective Projection : 원근법의 원리로써 Viewing Vol이 정의되고 내부에 있는 물체만이 스크린에 투영된다. Vol밖에 있는 물체는 잘려나간다.

- Aspect ratio = w : h (w/h) : 카메라로 볼 수 있는 면적의 폭과 높이의 비율
- View angle : 시각
- Field of View : 일정한 거리에 카메라를 고정시킬 때, 볼 수 있는 면적의 폭

VCS에 대한 점 P_v의 좌표가 (x_w, y_w, z_w)일 경우 Perspective Projection에 의해 생기는 스크린의 좌표 $P_s(x_s, y_s)$는

$$x_s = f \cdot x_v/(L - z_v)$$
$$y_s = f \cdot y_v/(L - z_v)$$
$$\begin{bmatrix} x_s \\ y_s \\ z_s \\ 1 \end{bmatrix} = \begin{bmatrix} 1\,0 & 0 & 0 \\ 0\,1 & 0 & 0 \\ 0\,0 & 1 & 0 \\ 0\,0 & -\frac{1}{f} & \frac{L}{f} \end{bmatrix} \begin{bmatrix} x_v \\ y_v \\ z_v \\ 1 \end{bmatrix}$$

7) Viewport Transformation

모델 형상의 점 및 좌표는 다음과 같은 순서로 변환되어 화면에 표시된다.

MCS -> (WCS) -> VCS -> SCS

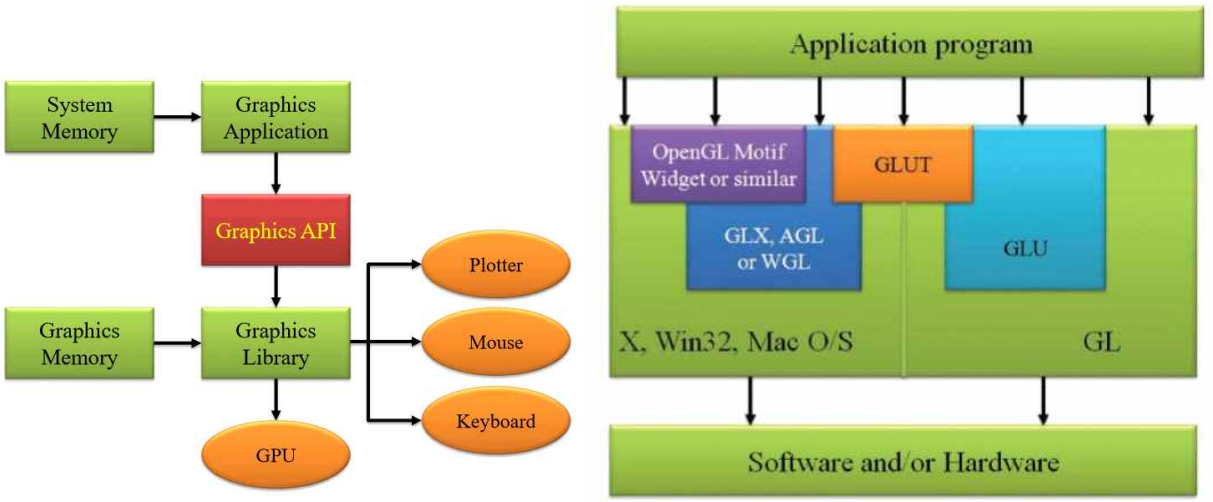
for example, vertexShader 코드상에서 시현된 좌표변환 코드.

```
uniform mat4 mM; //WCS or MCS
uniform mat4 mV; //VCS
unifomr mat4 mP; //SCS

in vec4 s_vColor;
out vec4 color;

void main() {
```

```
        color = s_vColor; //받은 색상좌표값을 내보낸다.
        gl_Position = mP*mV*mM*s_vPosition /* 좌표값을 MCS->VCS->SCS순으로 곱해나간다.*/
}
```

<openGL의 기본 개념>

1) openGL의 컴퓨터상의 상대적 위치 및 계층구조



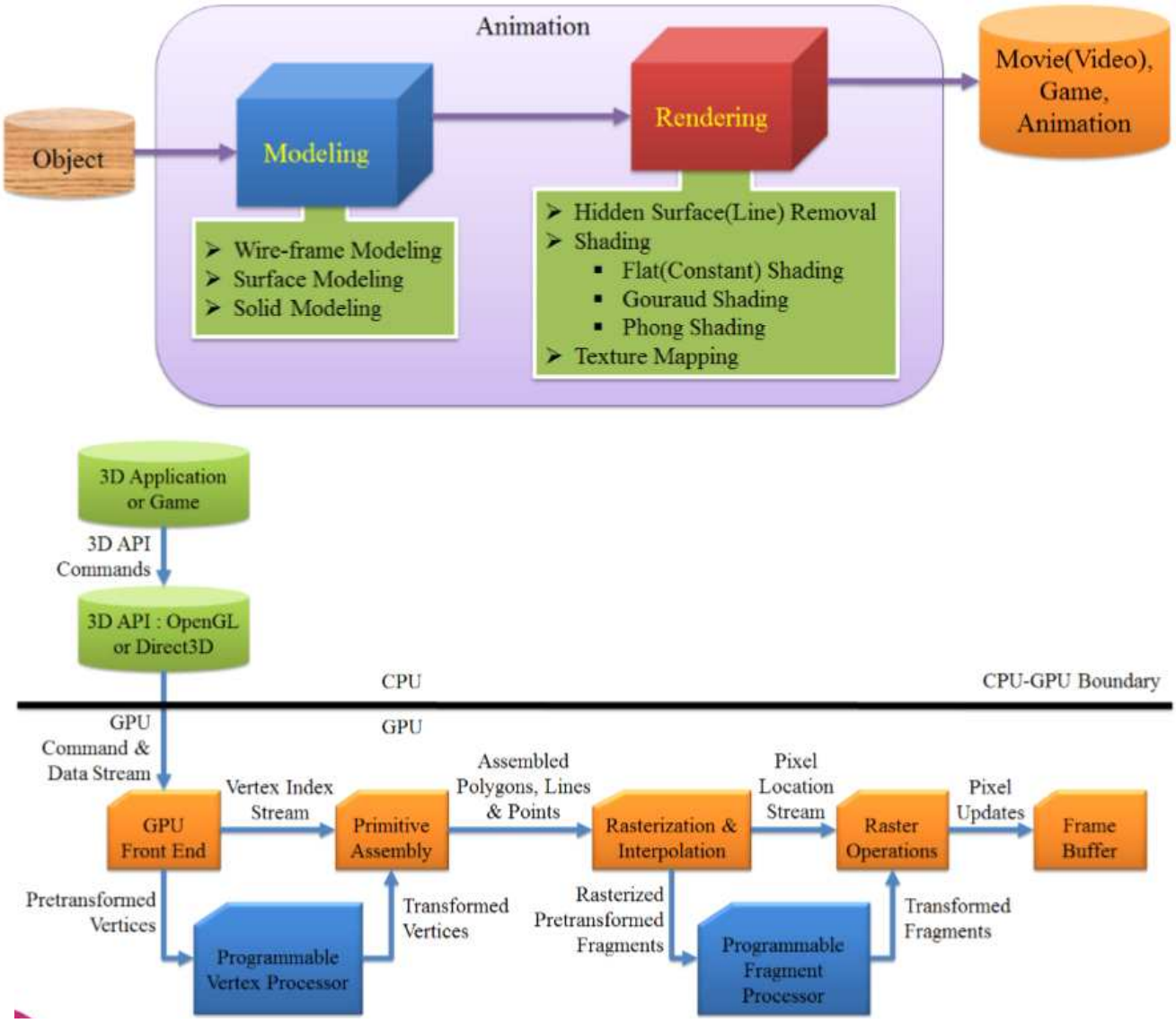
2) 주로 많이 사용되는 openGL -> GLUT의 함수들(Windows기능)

Function Name		Description
Window 초기화	glutInit();	OS와 Session 연결 / GLUT Lib를 초기화
	glutInitWindowPosition();	Monitor에서 Window의 시작점 위치 설정
	glutInitWindowSize();	Window의 크기(해상도) 설정
	glutInitDisplayMode();	Display Mode 설정
Window 관리	glutSetWindowTitle();	Window Title 설정
	glutCreateWindow();	새로운 Window창 생성
	glutReshapeWindow	크기 변경에 따른 Window 조정
	glutMainloop();	GLUT Event 처리 Loop 입력

3) 주로 많이 사용되는 openGL -> GLUT의 함수들(callback 기능)

Function Name	Description
glutDisplayFunc()	현재 Window를 위한 Display Callback을 설정한다.
glutReshapeFunc()	현재 Window를 위한 Reshape Callback을 설정한다.
glutKeyboardFunc()	현재 Window를 위한 Keyboard Callback을 설정한다.
glutSpecialFunc()	현재 Window를 위한 Special Keyboard Callback을 설정한다.
glutMouseFunc()	현재 Window를 위한 Mouse Callback을 설정한다.
glutMotionFunc()	현재 Window를 위한 각각의 Motion Callback을 설정한다.
glutPassiveMotionFunc()	현재 Window를 위한 Passive Motion Callback을 설정한다.
glutEntryFunc()	현재 Window를 위한 Mouse Enter/Leave Callback을 설정한다.
glutMouseWheelFunc()	현재 Window를 위한 Mouse Wheel Callback을 설정한다.
glutCreateMenu()	새로운 팝업 메뉴를 생성한다.
glutSetMenu()	현재 메뉴를 생성한다.
glutAddMenuEntry()	현재 메뉴의 하단에 메뉴항목을 추가한다.
glutAttachMenu()	현재 메뉴의 식별자로 현재 Window를 위한 MouseButton을 추가
glutAddSubMenu()	현재 메뉴의 하단에 서브 메뉴 Trigger를 추가한다.
glutIdleFunc()	현재 Window를 위한 Idle CallBack을 설정한다.
glutTimerFunc()	Milliseconds의 지정된 숫자로 Trigger되는 Timer Callback을 설정

4) openGL의 Graphic Pipeline And Real-Time Pipeline



참고문헌

(1) Radial engine-Wikipedia, https://en.wikipedia.org/wiki/Radial_engine

(2) 〈Design a four-cylinder Internal Combution Engine〉, Editor: Radoslav Plamenov Georgiev, 2011

(3) 〈Design a Radial Engine〉 Editor: M Tsankov Vasilev, 2011

(4) Radial Engine-Wikipedia, namu.wiki/w/성형엔진

(5) CAD/CAM강의자료 - 명지대학교 기계공학과

(6) openGL 강의 - http://3d.sangji.ac.kr/ppt/CG/