MÔN HỌC: ĐỒ HỌA MÁY TÍNH

LỚP CQ2016/2 – Thầy: **Lý Quốc Ngọc**

SEMINAR ĐỒ ÁN MÔN HỌC

Xây dựng ứng dụng 3D (tĩnh và động) dựa vào OpenGL trên môi trường Windows.

Đặng Phương Nam – Tạ Đăng Hiếu Nghĩa – Trần Bá Ngọc

1

Nhóm AutoPass

NỘI DUNG

1. SỬ DỤNG OPENGL CHO CÁC TÁC VỤ 3D.
2. XÂY DỰNG ỨNG DỤNG.

OPENGL CHO 3D

OpenGL là bộthư viện đồ họa có khoảng 150 hàm giúp xây dựng các đối tượng và giao tác cần thiết trong các ứng dụng tương tác 3D.

OpenGL hỗ trợ:

- ✓ Xây dựng các đối tượng phức tạp từ các đôi tượng hình học cơ bản.
- ✓ Sắp đối tượng trong 3D và chọn điểm thuận lợi để quan sát.
- ✓ Tính toán màu sắc đối tượng
- ✓ Biến đổi những mô tả về toán học của đối tượng.
- ✓ ….

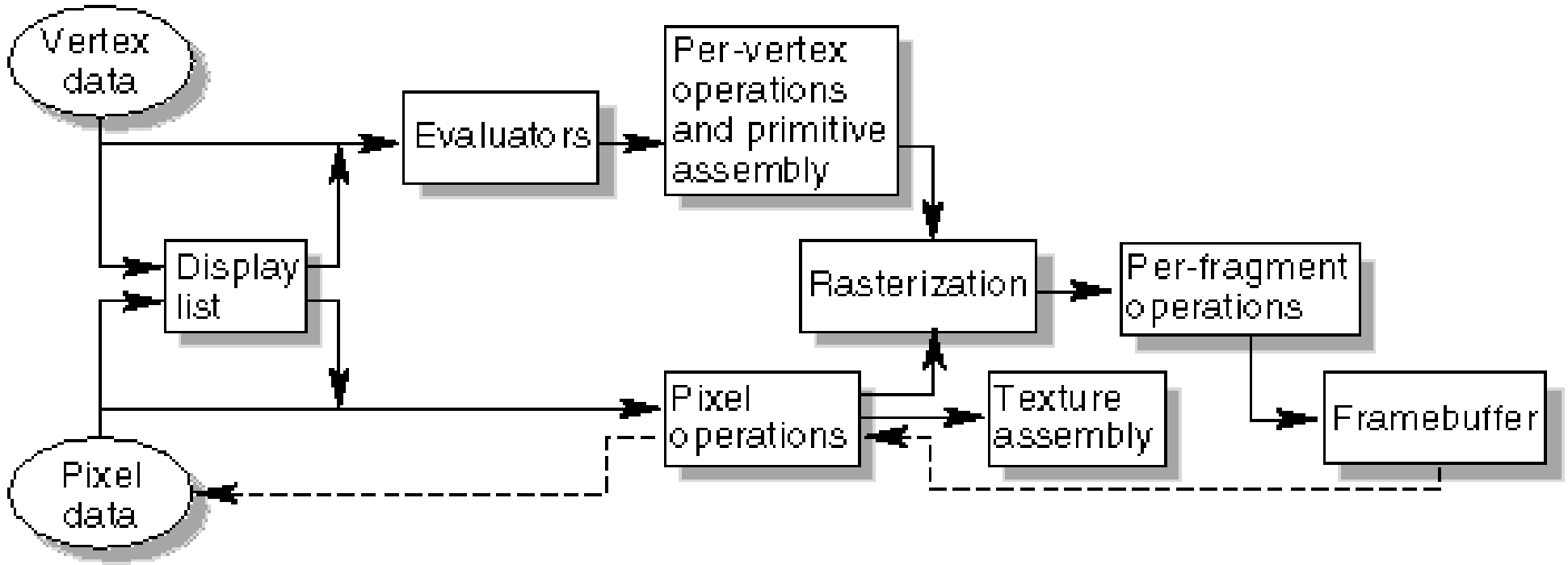
OpenGL không hỗ trợ:

- ❖ Không có hàm nhập xuất hay thao tác trên Window.
- ❖ Không có các hàm cấp cao để xây dựng mô hình đối tượng.

-> GLUT (OpenGL Utility Toolkit) là một thư viên được xây dựng để khắc phục những điều trên.

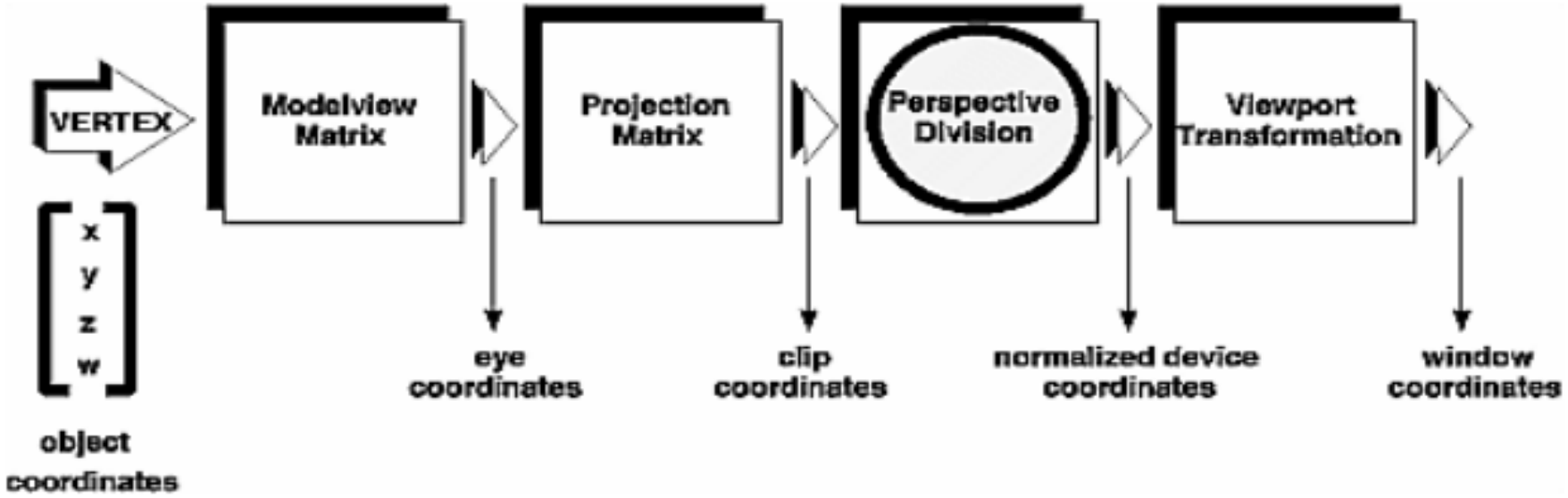
OPENGL CHO 3D

Cơ chế hoạt động OpenGL



OPENGL CHO 3D

Quy trình OpenGL biến đổi 1 điểm 3D trong thế giới thực thành 1 pixel trên màn hình:



OPENGL CHO 3D

Thao tác trên ModelView

glMatrixMode(GL_MODELVIEW);

***Biến đổi affine*:**

//tịnh tiến

glTranslate{fd}(TYPE x, TYPE y, TYPEz);

//quay quanh trục nối gốc tọa độ với điểm (x,y,z)

glRotate{fd}(TYPE angle, TYPE x, TYPE y, TYPE z);

//tỉ lệ (tâm tỉ lệ tại gốc tọa độ)

glScale{fd}(TYPEx, TYPE y, TYPEz);

Thiết lập view:

```
void gluLookAt(GLdouble eyex, GLdouble eyey, GLdouble eyez, GLdouble
centerx, GLdouble centery, GLdouble centerz, GLdouble upx, GLdouble
upy, GLdouble upz)
```

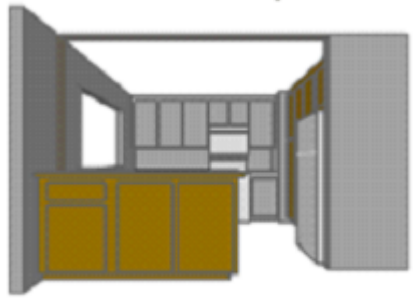
OPENGL CHO 3D

Thao tác trên Projection

glMatrixMode(GL_PROJECTION);

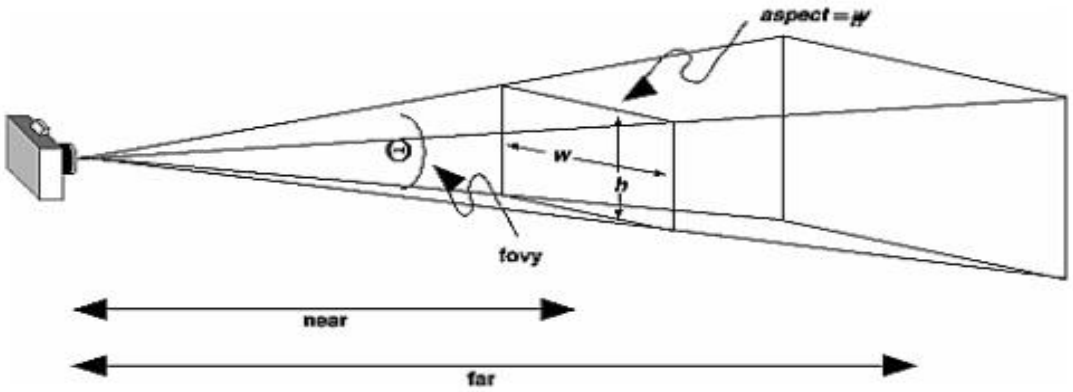
```
void gluPerspective(GLdouble fovy, GLdouble aspect, GLdouble near,
GLdouble far);
```

trong đó các tham số được miêu tả như hình dưới đây



Perspective projection

Chiếu phối cảnh



(aspect = w/h).

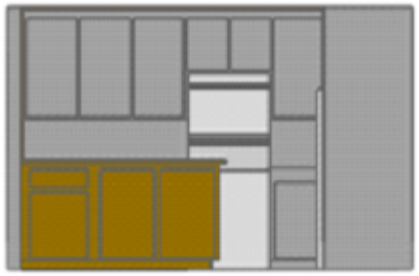
OPENGL CHO 3D

Thao tác trên Projection

glMatrixMode(GL_PROJECTION);

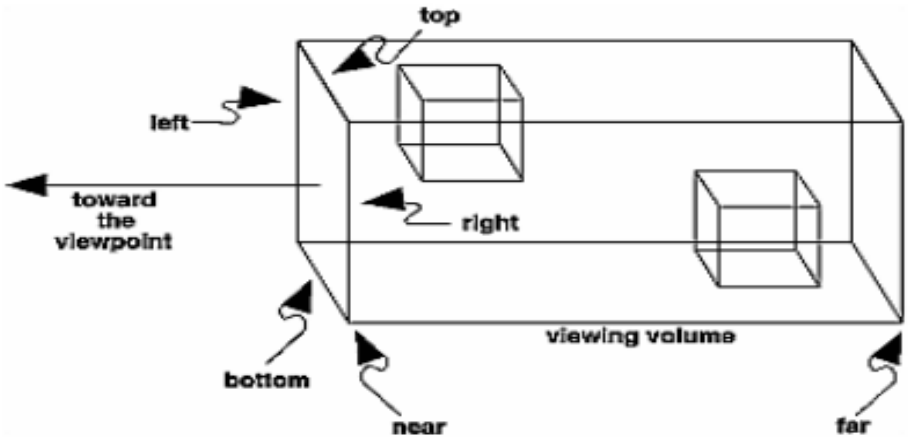
```
void glOrtho(GLdouble left, GLdouble right, GLdouble bottom, GLdouble
top, GLdouble near, GLdouble far);
```

trong đó các tham số được thể hiện trong hình dưới đây.



Parallel projection

Chiếu trực giao



OPENGL CHO 3D

Thao tác trên ViewPort:

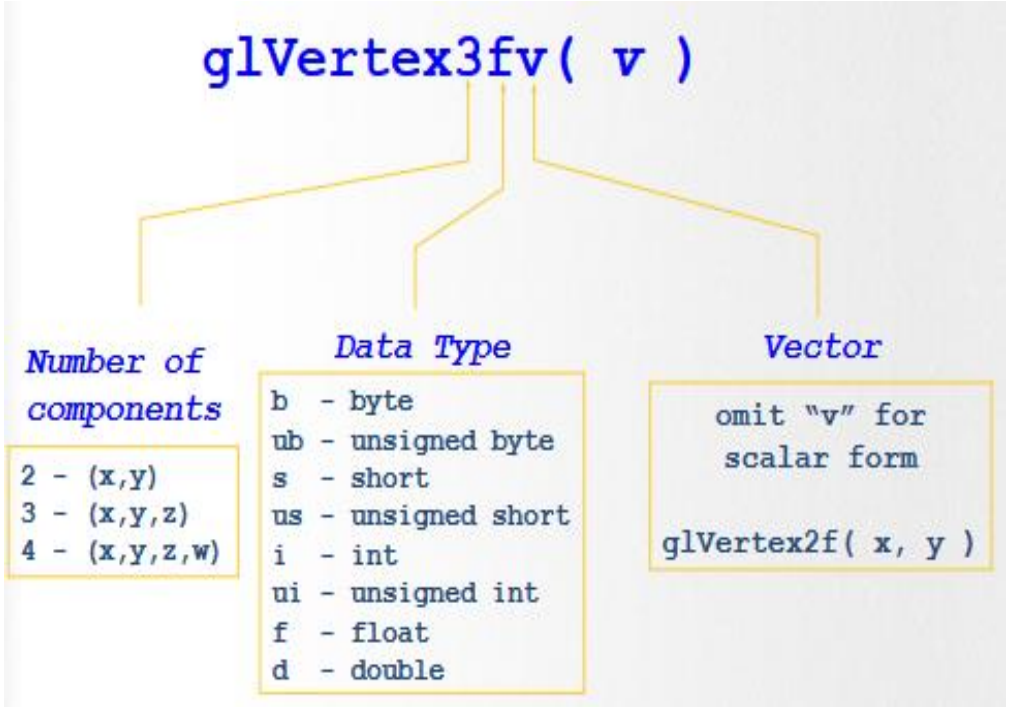
Hàm thiết lập viewport:

```
void glViewport(GLint x, GLint y, GLsizei width, GLsizei height);
```

- (x,y) là vị trí điểm trái-trên trong cửa sổ vẽ, width, height là chiều rộng và cao của viewport.
- (x,y,width,height) =(0,0,winWidth, winHeight) (chiếm toàn bộ cửa sổ)

OPENGL CHO 3D

Cấu trúc lệnh trong OpenGL

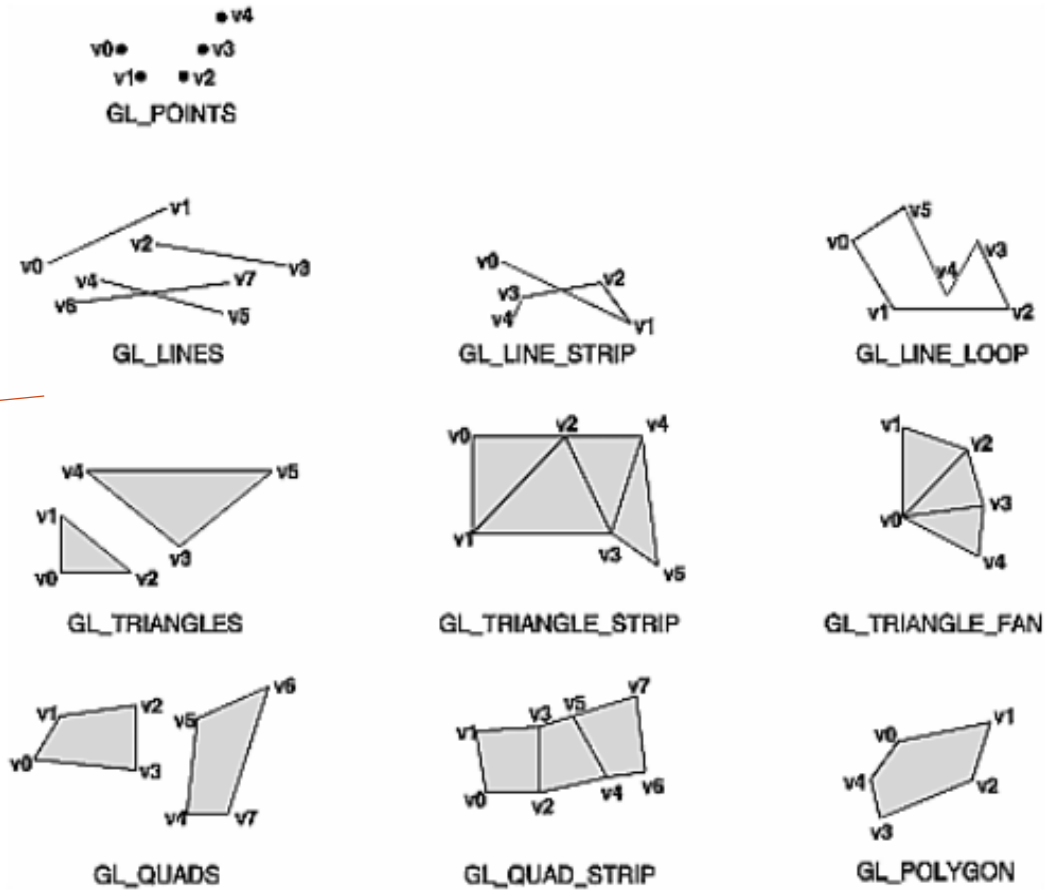


Hằng số trong Opengl: OpenGL đặt tên các hằng số bắt đầu bằng GL_và các từ tiếp sau đều được viết hoa và cách nhau bởi dấu '_',
Ví dụ:GL_COLOR_BUFFER_BIT.

OPENGL CHO 3D

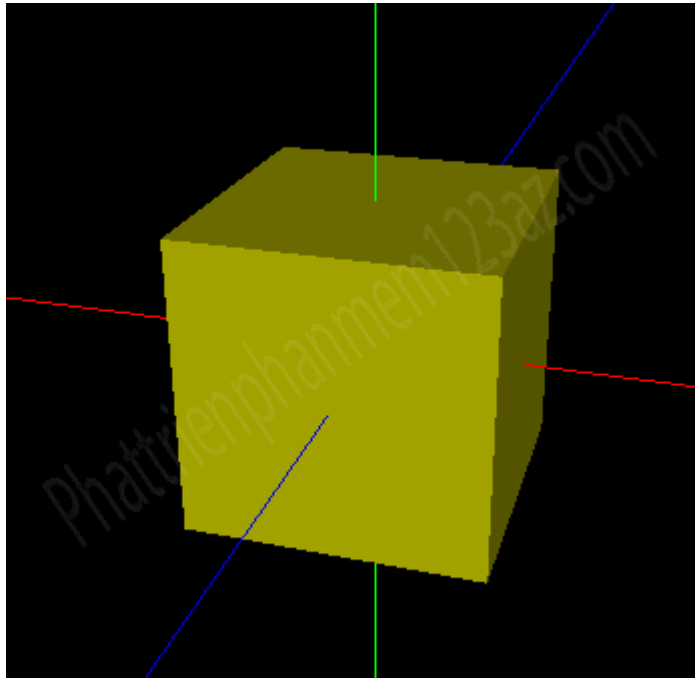
Vẽ đối tượng cơ bản

```
glColor3f(1.0,0.0,0.0); //màu vẽ

glBegin(mode);
/* xác định tọa độ các điểm cần vẽ*/
glEnd();

glFlush(); //render ngay ra màn hình
```



OPENGL CHO 3D

Ví dụ vẽ hình Cube 3D



```
int size =10;

glBegin(GL_QUADS);
 // Vẽ mặt trước
 glVertex3f(-size, -size, size);
 glVertex3f( size, -size, size);
 glVertex3f( size, size, size);
 glVertex3f(-size, size, size);

 // Vẽ mặt sau
 // Vẽ mặt trên
 // Vẽ mặt dưới
 // Vẽ mặt bên trái
 // Vẽ mặt bên phải

 glEnd();
 glFlush();
```

OPENGL CHO 3D

GLUT có hỗ trợ sẵn một số hàm để vẽ các đối tượng hình học phức tạp hơn:

```
void glutWireSphere(GLdouble radius, GLint slices, GLint stacks);
void glutSolidSphere(GLdouble radius, GLint slices, GLint stacks);
void glutWireCube(GLdouble size);
void glutSolidCube(GLdouble size);
void glutWireTorus(GLdouble innerRadius, GLdouble outerRadius, GLint nsides,
GLint rings);
void glutSolidTorus(GLdouble innerRadius, GLdouble outerRadius, GLint nsides,
GLint rings);
void glutWireIcosahedron(void);
void glutSolidIcosahedron(void);
void glutWireOctahedron(void);
void glutSolidOctahedron(void);
void glutWireTetrahedron(void);
void glutSolidTetrahedron(void);
void glutWireDodecahedron(GLdouble radius);
void glutSolidDodecahedron(GLdouble radius);
void glutWireCone(GLdouble radius, GLdouble height, GLint slices, GLint
stacks);
void glutSolidCone(GLdouble radius, GLdouble height, GLint slices, GLint
stacks);
void glutWireTeapot(GLdouble size);
void glutSolidTeapot(GLdouble size);
```

OPENGL CHO 3D

OpenGL còn cung cấp một số xử lý sau:

- ✓ Chiếu sáng.
- ✓ Display List.
- ✓ Khử răng cưa, hiệu ứng sương mù.
- ✓ Dán Texture.
- ✓ ….

XÂY DỰNG ỨNG DỤNG

1. Giới thiệu
2. Phát biểu bài toán
3. Giải quyết bài toán
4. Thực nghiệm
5. Kết luận

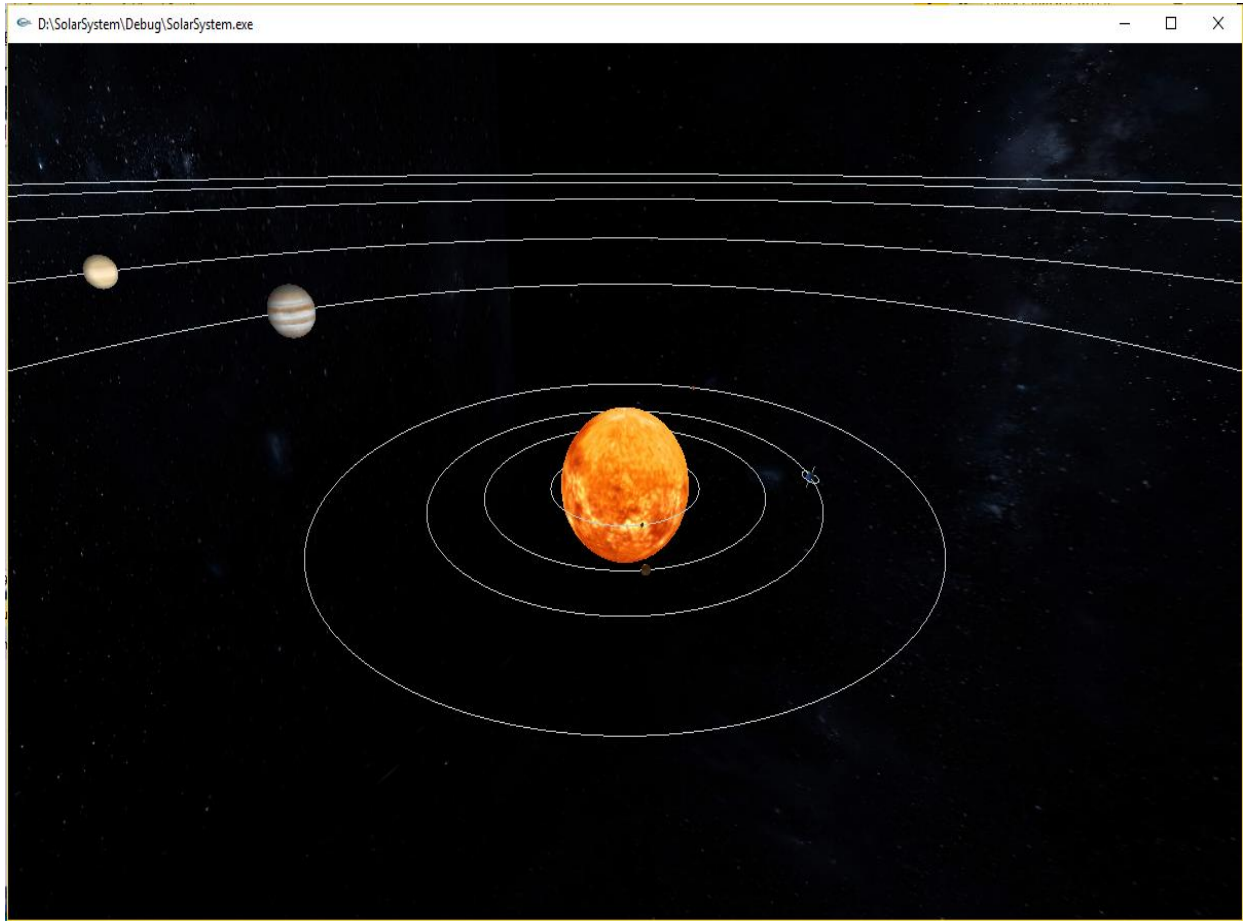
GIỚI THIỆU

- Rất nhiều sản phẩm của công nghệ đã được áp dụng vào đời sống

- Việc áp dụng công nghệ đã giúp ngành giáo dục trở nên phát triển hơn, giúp có cái nhìn trực quan hơn về thực tế

⇒ Từ những nhu cầu thực tế + những kiến thức đã học về Đồ hoạ máy tính -> "Mô phỏng hệ mặt trời 3D"

⇒ Có thể cung cấp các kiến thức cơ bản về vũ trụ, giúp học sinh có cách tiếp cận tốt hơn với vũ trụ, cũng như giúp giáo viên dễ dàng hơn trong việc truyền đạt kiến thức



PHÁT BIỂU BÀI TOÁN

- **Mục đích :**
  - Vận dụng các kiến thức đã tìm hiểu về OpenGL cho đối tượng 3D để xây dựng nên một ứng dụng cơ bản nhưng hữu ích.
  - Tạo ra một mô hình 3D về Hệ Mặt Trời, với các kiến thức phổ thông đã được học để mô phỏng chính xác, chi tiết
- **Lợi ích:**
  - Là một mô hình demo về Hệ Mặt Trời có thể dùng để giảng dạy trong trường học.
  - Có thể được dùng làm một tài liệu tham khảo về mô phỏng vũ trụ.
  - Cung cấp cho mọi người có cái nhìn trừu tượng về Hệ Mặt Trời.

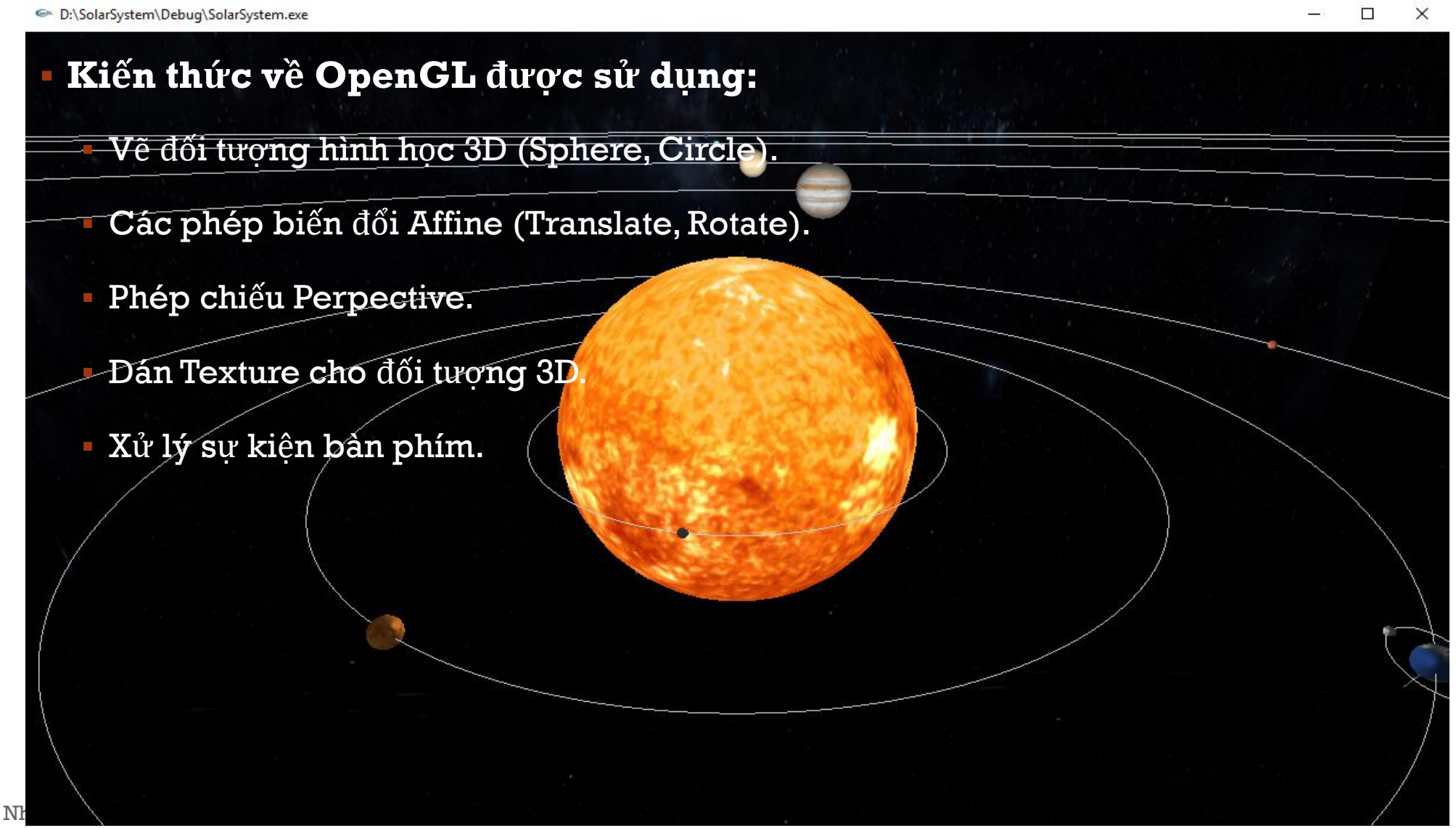
PHÁT BIỂU BÀI TOÁN

- **Giới hạn bài toán:**
  - Ứng dụng có khả năng hiển thị tương đối về: đường kính, khoảng cách, quỹ đạo quay, và tốc độ quay của các hành tinh xung quanh Mặt Trời.
  - Ứng dụng phải thể hiện đúng quỹ đạo quay của Trái Đất (tự quanh mình nghiên một góc 24,3°) và phải có Mặt Trăng quay xung quanh.
  - Ứng dụng phải dán texture cho các hành tinh để có cái nhìn trực quan sinh động về Hệ Mặt Trời, đồng thời cung cấp một số thông tin thú vị về các hành tinh để thu hút người dùng.
  - Ứng dụng có khả năng tương tác người dùng (thông qua bàn phím).
  - Ứng dụng được thiết kế dựa vào API OpenGL, ngôn ngữ C++.

GIẢI QUYẾT BÀI TOÁN

- **Hệ Mặt Trời:**
  - Thông tin Mặt Trời, thứ tự của các hành tinh được sắp xếp tăng dần theo khoảng cách đến Mặt Trời và quỹ đạo quay của chúng (bao gồm quỹ đạo quay quanh Mặt Trời và quỹ đạo tự quay 360° quanh mình).
  - Hiển thị kích thước tương đối giữa các hành tinh so với kích thước thực tế.
  - Khoảng cách từ hành tinh đến Mặt Trời: Hiển thị khoảng cách tương đối giữa các hành tinh so với tốc độ thực tế.
  - Tốc độ quay của hành tinh: tốc độ tương đối so với thực tế.
  - Quỹ đạo quay của hành tinh: Hiển thị mọi quỹ đạo trong Hệ Mặt Trời.
  - Chế độ chiếu sáng: Mặt Trời sẽ là nguồn sáng duy nhất trong hệ.

GIẢI QUYẾT BÀI TOÁN



- **Kiến thức về OpenGL được sử dụng:**
  - Vẽ đối tượng hình học 3D (Sphere, Circle).
  - Các phép biến đổi Affine (Translate, Rotate).
  - Phép chiếu Perpective.
  - Dán Texture cho đối tượng 3D.
  - Xử lý sự kiện bàn phím.

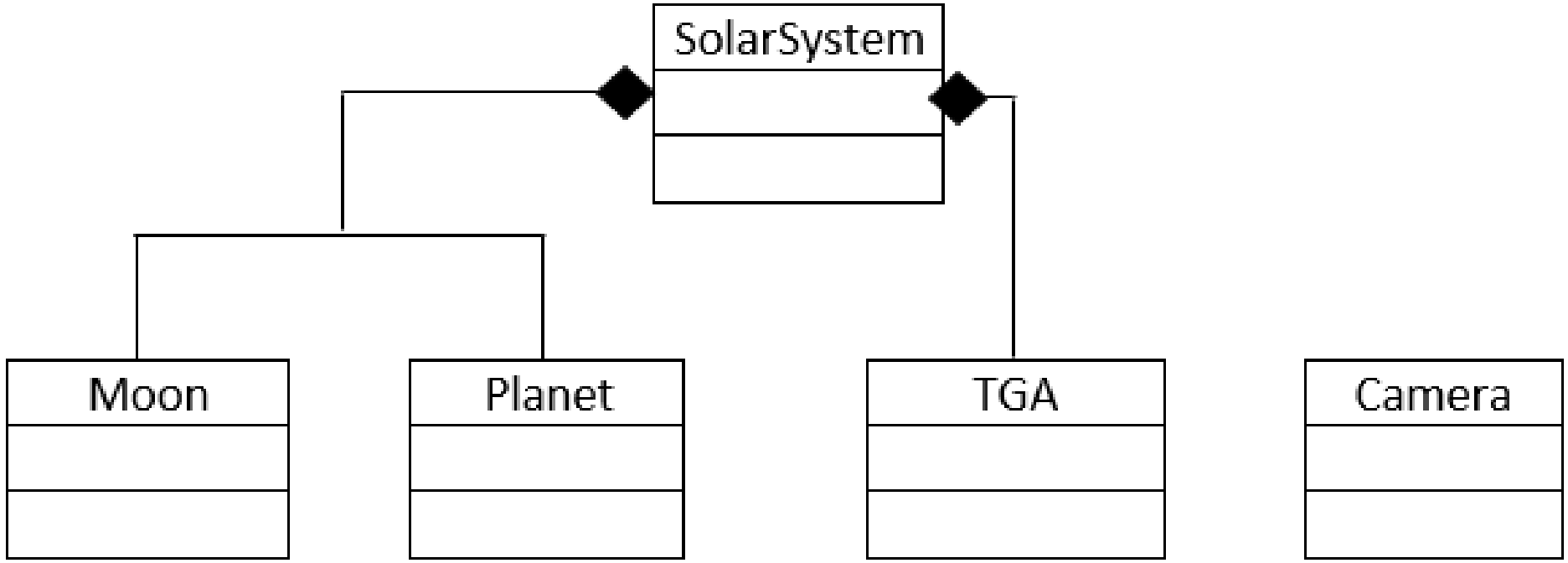
GIẢI QUYẾT BÀI TOÁN

Các công đoạn tạo hệ mặt trời:

- Khoảng cách từ hành tinh tới Mặt Trời, bán kính, tốc độ quay
- Khai báo các giá trị tỉ lệ thực so với trên ứng dụng
- Tạo Mặt Trời
- Tạo các hành tinh khác.
- Xác định tốc độ quay cho mỗi hành tinh
- Vẽ them mặt trăng
- Tạo quỹ đạo quay của các hành tinh: quay quanh mặt trời + quay quanh trục
- Tạo thêm chức năng bàn phím để điều chỉnh chuyển động của camera
- Load texture cho các hành tinh để tạo sự chân thật cho ứng dụng.
- Điều chỉnh ánh sáng để tạo hiệu ứng 3D.
- Cập nhật trạng thái của toàn bộ Hệ Mặt Trời sau những khoảng thời gian khác nhau.
- Xử lý các sự kiện bàn phím từ người dùng.

GIẢI QUYẾT BÀI TOÁN

Sơ đồ lớp ứng dụng:[5]



TÀI LIỆU THAM KHẢO

- [1] Joey de Vries, Learn OpenGL, An offline transcript of learnopengl.com, 2015
- [2] Mark Segal, Kurt Akeley, The OpenGLR Graphics System, The Khronos Group, 2010
- [3] https://pds.jpl.nasa.gov/planets/
- [4] https://www.solarsystemscope.com/textures/
- [5] https://github.com/RyanPridgeon/solarsystem
- [6] https://thuthuattienich.vn/thu-thuat/opengl-la-gi
- [7] http://glprogramming.com/red/