Grafica pe calculator. Preliminarii

Mihai-Sorin Stupariu

Sem. I, 2022 - 2023

Motivație

Aspecte introductive

Despre OpenGL

Exemple de programe

De ce un curs de grafică? / Aplicații?

- Filme.

De ce un curs de grafică? / Aplicații?

- Filme.
- Dezvoltarea de jocuri.

De ce un curs de grafică? / Aplicații?

- Filme.
- Dezvoltarea de jocuri.
- Imagistică medicală.

Ce înseamnă un curs de grafică?

- **Abordări posibile:**

Ce înseamnă un curs de grafică?

- **Abordări posibile:**
  - background + dezvoltare "low level" (de exemplu *OpenGL*)

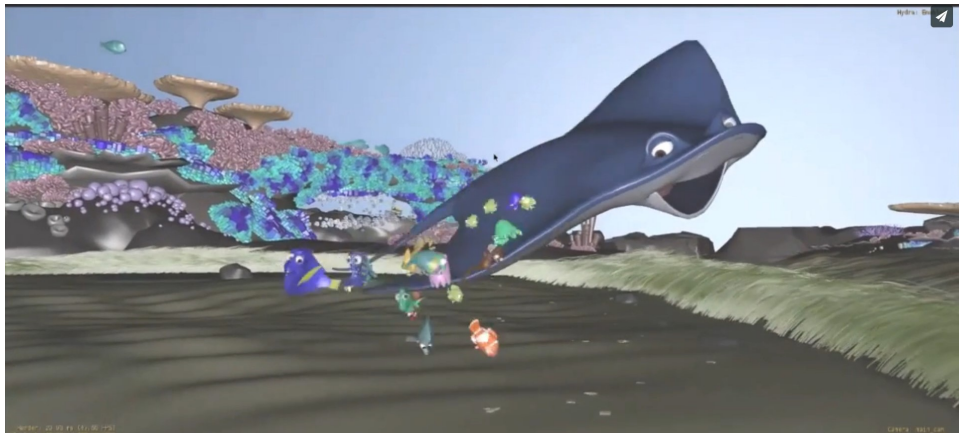


Figura: Sursa: https://www.iamag.co/real-time-graphics-in-pixar-film-production/

Ce înseamnă un curs de grafică?

- **Abordări posibile:**
  - background + dezvoltare "low level" (de exemplu *OpenGL*)

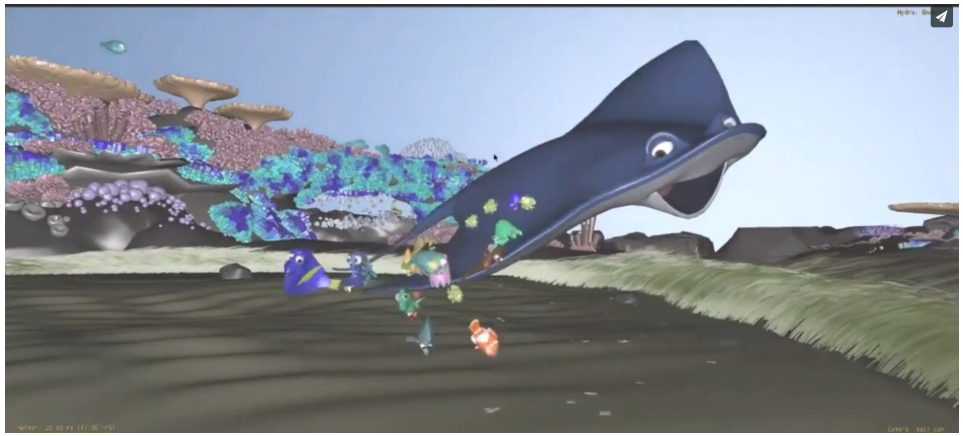


Figura: Sursa: https://www.iamag.co/real-time-graphics-in-pixar-film-production/

  - tehnologii dedicate (de exemplu *Unity*)

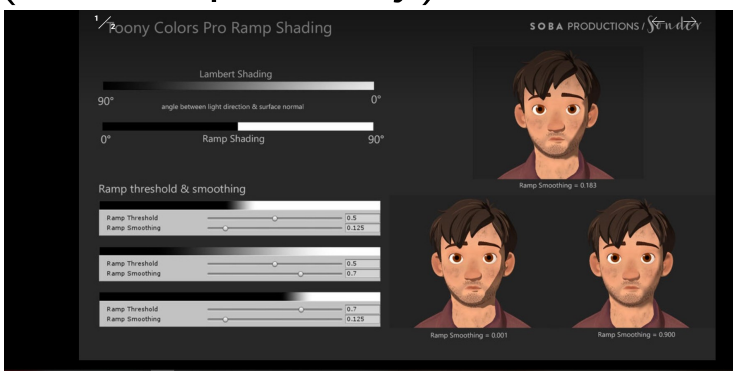


Figura: Sursa: https://unity.com/madewith/sonder

Cunoștințe necesare?

- cunoștințe elementare de programare în C++ (inclusiv utilizarea unui mediu de dezvoltare integrat) - vom folosi Microsoft Visual Studio;
- elemente de bază de Algebră liniară și Geometrie analitică.

Ce teme vom aborda la curs+laborator? Care ar fi obiectivul?

- Primitive grafice

Ce teme vom aborda la curs+laborator? Care ar fi obiectivul?

- Primitive grafice
- Transformări

Ce teme vom aborda la curs+laborator? Care ar fi obiectivul?

- Primitive grafice
- Transformări
- Texturare

Ce teme vom aborda la curs+laborator? Care ar fi obiectivul?

- Primitive grafice
- Transformări
- Texturare
- Reprezentarea scenelor 3D

Ce teme vom aborda la curs+laborator? Care ar fi obiectivul?

- Primitive grafice
- Transformări
- Texturare
- Reprezentarea scenelor 3D
- Iluminare

Ce teme vom aborda la curs+laborator? Care ar fi obiectivul?

- Primitive grafice
- Transformări
- Texturare
- Reprezentarea scenelor 3D
- Iluminare
- Efecte vizuale

Ce teme vom aborda la curs+laborator? Care ar fi obiectivul?

- Primitive grafice
- Transformări
- Texturare
- Reprezentarea scenelor 3D
- Iluminare
- Efecte vizuale
- Ideal: la finalul cursului să puteți dezvolta o aplicație complexă 3D

Resurse

- Site-ul oficial OpenGL

Resurse

- Site-ul oficial OpenGL
- Cărți:
  - D. Hearn si M. Baker, *Computer Graphics with OpenGL*, 2003
  - D. Shreiner, M.Woo, J. Neider si T. Davis. *OpenGL Programming Guide* (2nd edition), Addison Wesley, 1997
  - D. Shreiner, G. Sellers, J. Kessenich, B. Licea-Kane *OpenGL Programming Guide* (9th edition), Addison Wesley, 2016
  - G. Sellers, R. S. Wright Jr., N. Haemel, *OpenGL: SuperBible. Comprehensive Tutorial and Reference* (7th edition), Addison-Wesley, 2015,

Resurse

- Site-ul oficial OpenGL
- Cărți:
  - D. Hearn si M. Baker, *Computer Graphics with OpenGL*, 2003
  - D. Shreiner, M.Woo, J. Neider si T. Davis. *OpenGL Programming Guide* (2nd edition), Addison Wesley, 1997
  - D. Shreiner, G. Sellers, J. Kessenich, B. Licea-Kane *OpenGL Programming Guide* (9th edition), Addison Wesley, 2016
  - G. Sellers, R. S. Wright Jr., N. Haemel, *OpenGL: SuperBible. Comprehensive Tutorial and Reference* (7th edition), Addison-Wesley, 2015,
- Cărți / tutoriale online:
  - http://learnopengl.com/
  - http://www.opengl-tutorial.org/
  - D. Eck, Introduction to Computer Graphics
  - http://nehe.gamedev.net/
  - A. Gerdelan - OpenGL 4 tutorials;
  - etc.

Elemente generale

- OpenGL este o interfață de programare (Application Programming Interface API)

Elemente generale

- OpenGL este o interfață de programare (Application Programming Interface API)
- Pentru a funcționa are nevoie de o serie de biblioteci; versiunea suportată depinde de placa grafică a calculatorului - se poate folosi codul sursa `00_test_version.cpp`

Elemente generale

- OpenGL este o interfață de programare (Application Programming Interface API)
- Pentru a funcționa are nevoie de o serie de biblioteci; versiunea suportată depinde de placa grafică a calculatorului - se poate folosi codul sursa `00_test_version.cpp`
- OpenGL **nu** este un limbaj de programare (există GLSL)

Elemente generale

- OpenGL este o interfață de programare (Application Programming Interface API)
- Pentru a funcționa are nevoie de o serie de biblioteci; versiunea suportată depinde de placa grafică a calculatorului - se poate folosi codul sursa `00_test_version.cpp`
- OpenGL **nu** este un limbaj de programare (există GLSL)
- Arhitectura de tip *client-server* (CPU-GPU) - element cheie: "comunicarea" dintre cele două componente *hardware*

Scurt istoric

- 1992: versiunea 1.0 a OpenGL (derivat din IRIS GL), lansat de SGI; “open standard”

Scurt istoric

- 1992: versiunea 1.0 a OpenGL (derivat din IRIS GL), lansat de SGI; “open standard”
- 1992: OpenGL Architecture Review Board (elaborarea specificațiilor)

Scurt istoric

- 1992: versiunea 1.0 a OpenGL (derivat din IRIS GL), lansat de SGI; "open standard"
- 1992: OpenGL Architecture Review Board (elaborarea specificațiilor)
- 1997: versiunea 1.1

Scurt istoric

- 1992: versiunea 1.0 a OpenGL (derivat din IRIS GL), lansat de SGI; "open standard"
- 1992: OpenGL Architecture Review Board (elaborarea specificațiilor)
- 1997: versiunea 1.1
- 2002/2003: versiunile 1.4, 1.5; GLSL (GL Shading Language) apare ca o extensie a funcționalității de bază

Scurt istoric

- 1992: versiunea 1.0 a OpenGL (derivat din IRIS GL), lansat de SGI; "open standard"
- 1992: OpenGL Architecture Review Board (elaborarea specificațiilor)
- 1997: versiunea 1.1
- 2002/2003: versiunile 1.4, 1.5; GLSL (GL Shading Language) apare ca o extensie a funcționalității de bază
- 2004: **OpenGL 2.0; GLSL 1.10.59 inclus în "core"**

Scurt istoric

- 1992: versiunea 1.0 a OpenGL (derivat din IRIS GL), lansat de SGI; "open standard"
- 1992: OpenGL Architecture Review Board (elaborarea specificațiilor)
- 1997: versiunea 1.1
- 2002/2003: versiunile 1.4, 1.5; GLSL (GL Shading Language) apare ca o extensie a funcționalității de bază
- 2004: **OpenGL 2.0; GLSL 1.10.59 inclus în "core"**
- 2008: OpenGL 3.0: conceptul de "funcționalități depreciate", legate de modul de redare imediat; mecanismul de depreciere a fost implementat / extins în 2009, când au fost lansate versiunile 3.1 și 3.2

Scurt istoric

- 1992: versiunea 1.0 a OpenGL (derivat din IRIS GL), lansat de SGI; "open standard"
- 1992: OpenGL Architecture Review Board (elaborarea specificațiilor)
- 1997: versiunea 1.1
- 2002/2003: versiunile 1.4, 1.5; GLSL (GL Shading Language) apare ca o extensie a funcționalității de bază
- 2004: **OpenGL 2.0; GLSL 1.10.59 inclus în "core"**
- 2008: OpenGL 3.0: conceptul de "funcționalități depreciate", legate de modul de redare imediat; mecanismul de depreciere a fost implementat / extins în 2009, când au fost lansate versiunile 3.1 și 3.2
- 2010: OpenGL 3.3 (hardware care suportă Direct3D 10); OpenGL 4.0 (hardware care suportă Direct3D 11); numerotarea versiunilor de GLSL este "sincronizată" (GLSL v. 3.30.6, respectiv v. 4.00.9)

Scurt istoric

- 1992: versiunea 1.0 a OpenGL (derivat din IRIS GL), lansat de SGI; "open standard"
- 1992: OpenGL Architecture Review Board (elaborarea specificațiilor)
- 1997: versiunea 1.1
- 2002/2003: versiunile 1.4, 1.5; GLSL (GL Shading Language) apare ca o extensie a funcționalității de bază
- 2004: **OpenGL 2.0; GLSL 1.10.59 inclus în "core"**
- 2008: OpenGL 3.0: conceptul de "funcționalități depreciate", legate de modul de redare imediat; mecanismul de depreciere a fost implementat / extins în 2009, când au fost lansate versiunile 3.1 și 3.2
- 2010: OpenGL 3.3 (hardware care suportă Direct3D 10); OpenGL 4.0 (hardware care suportă Direct3D 11); numerotarea versiunilor de GLSL este "sincronizată" (GLSL v. 3.30.6, respectiv v. 4.00.9)
- 2014 OpenGL 4.5 (respectiv GLSL 4.50)

Scurt istoric

- 1992: versiunea 1.0 a OpenGL (derivat din IRIS GL), lansat de SGI; "open standard"
- 1992: OpenGL Architecture Review Board (elaborarea specificațiilor)
- 1997: versiunea 1.1
- 2002/2003: versiunile 1.4, 1.5; GLSL (GL Shading Language) apare ca o extensie a funcționalității de bază
- 2004: **OpenGL 2.0; GLSL 1.10.59 inclus în "core"**
- 2008: OpenGL 3.0: conceptul de "funcționalități depreciate", legate de modul de redare imediat; mecanismul de depreciere a fost implementat / extins în 2009, când au fost lansate versiunile 3.1 și 3.2
- 2010: OpenGL 3.3 (hardware care suportă Direct3D 10); OpenGL 4.0 (hardware care suportă Direct3D 11); numerotarea versiunilor de GLSL este "sincronizată" (GLSL v. 3.30.6, respectiv v. 4.00.9)
- 2014 OpenGL 4.5 (respectiv GLSL 4.50)
- 2017 OpenGL 4.6 (respectiv GLSL 4.60)

Biblioteci utilizate de OpenGL și funcții asociate

- bibliotecă fundamentală (core library): este independentă de platforma pe care se lucrează; funcțiile corespunzătoare au prefixul `gl`. De exemplu: (i) `glClearColor ( ); glFlush ( );` – comune; (ii) `glVertex ( ); glColor ( ); glBegin ( )` – depreciate; (iii) `glGenVertexArrays ( ); glDrawArrays ( )` – OpenGL nou.

Biblioteci utilizate de OpenGL și funcții asociate

- bibliotecă fundamentală (core library): este independentă de platforma pe care se lucrează; funcțiile corespunzătoare au prefixul `gl`. De exemplu: (i) `glClearColor ( ); glFlush ( );` – comune; (ii) `glVertex ( ); glColor ( ); glBegin ( )` – depreciate; (iii) `glGenVertexArrays ( ); glDrawArrays ( )` – OpenGL nou.
- GLU (OpenGL Utility): conține mai ales proceduri / funcții legate de proiecție, precum și funcții pentru conice și cuadrice; funcțiile asociate au prefixul `glu`

Biblioteci utilizate de OpenGL și funcții asociate

- bibliotecă fundamentală (core library): este independentă de platforma pe care se lucrează; funcțiile corespunzătoare au prefixul `gl`. De exemplu: (i) `glClearColor ( ); glFlush ( );` – comune; (ii) `glVertex ( ); glColor ( ); glBegin ( )` – depreciate; (iii) `glGenVertexArrays ( ); glDrawArrays ( )` – OpenGL nou.
- GLU (OpenGL Utility): conține mai ales proceduri / funcții legate de proiecție, precum și funcții pentru conice și cuadrice; funcțiile asociate au prefixul `glu`
- GLUT (OpenGL Utility Toolkit) / freeglut: bibliotecă *dependentă de sistem*, utilizată pentru a realiza fereastra de vizualizare; poate interacționa cu sisteme de operare bazate pe ferestre de vizualizare; funcțiile specifice au prefixul `glut`. Există și alte biblioteci / pachete, cum ar fi GLFW.

Exemplu de program care utilizează OpenGL “vechi”

Codul sursă `01_01_oldOpenGL.cpp`

```
// Directive preprocesare
```

Exemplu de program care utilizează OpenGL "vechi"

Codul sursă `01_01_oldOpenGL.cpp`

```
// Directive preprocesare
// Procedură inițializare – init
```

Exemplu de program care utilizează OpenGL "vechi"

Codul sursă `01_01_oldOpenGL.cpp`

```
// Directive preprocesare
// Procedură inițializare – init
// Procedură desen – desen
```

Exemplu de program care utilizează OpenGL "vechi"

Codul sursă `01_01_oldOpenGL.cpp`

```
// Directive preprocesare
// Procedură inițializare – init
// Procedură desen – desen
// Main
```

Exemplu de program care utilizează OpenGL "vechi"

Codul sursă `01_01_oldOpenGL.cpp`

```
// Directive preprocesare
// Procedură inițializare – init
// Procedură desen – desen
// Main
    // Inițializări GLUT
    // Generare fereastră
    // Apelare procedură inițializare
    // Apelare procedură desen
    // Apelare glutMainLoop
```

Exemplu de program care utilizează OpenGL “nou”

Codul sursă `01_02_newOpenGL.cpp`

```
// Directive preprocesare
```

Exemplu de program care utilizează OpenGL “nou”

Codul sursă `01_02_newOpenGL.cpp`

```
// Directive preprocesare
// Shader-e, date (coordonate, culori)
```

Exemplu de program care utilizează OpenGL “nou”

Codul sursă `01_02_newOpenGL.cpp`

```
// Directive preprocesare
// Shader-e, date (coordonate, culori)
// Proceduri (inițializare, desen)
```

Exemplu de program care utilizează OpenGL “nou”

Codul sursă `01_02_newOpenGL.cpp`

```
// Directive preprocesare
// Shader-e, date (coordonate, culori)
// Proceduri (inițializare, desen)
// Main
```

Exemplu de program care utilizează OpenGL "nou"

Codul sursă `01_02_newOpenGL.cpp`

```
// Directive preprocesare
// Shader-e, date (coordonate, culori)
// Proceduri (inițializare, desen)
// Main
    // Inițializări GLUT
    // Generare fereastră
    // Apelare procedură inițializare
        // Creare VBO / VAO
        // Creare / Apelare shader-e
```

Exemplu de program care utilizează OpenGL "nou"

Codul sursă `01_02_newOpenGL.cpp`

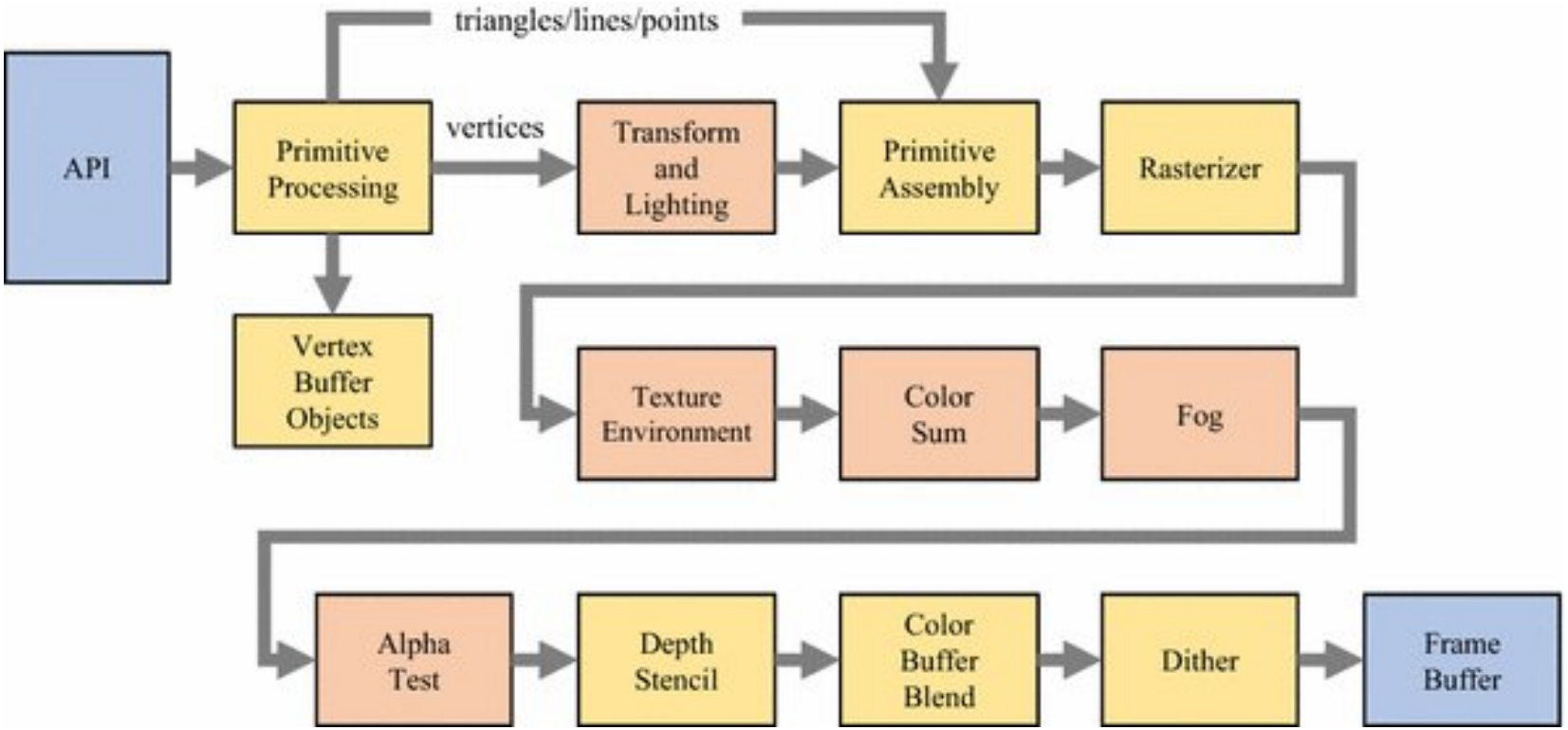
```
// Directive preprocesare
// Shader-e, date (coordonate, culori)
// Proceduri (inițializare, desen)
// Main
    // Inițializări GLUT
    // Generare fereastră
    // Apelare procedură inițializare
        // Creare VBO / VAO
        // Creare / Apelare shader-e
    // Apelare procedură desen
```

Exemplu de program care utilizează OpenGL "nou"

Codul sursă `01_02_newOpenGL.cpp`

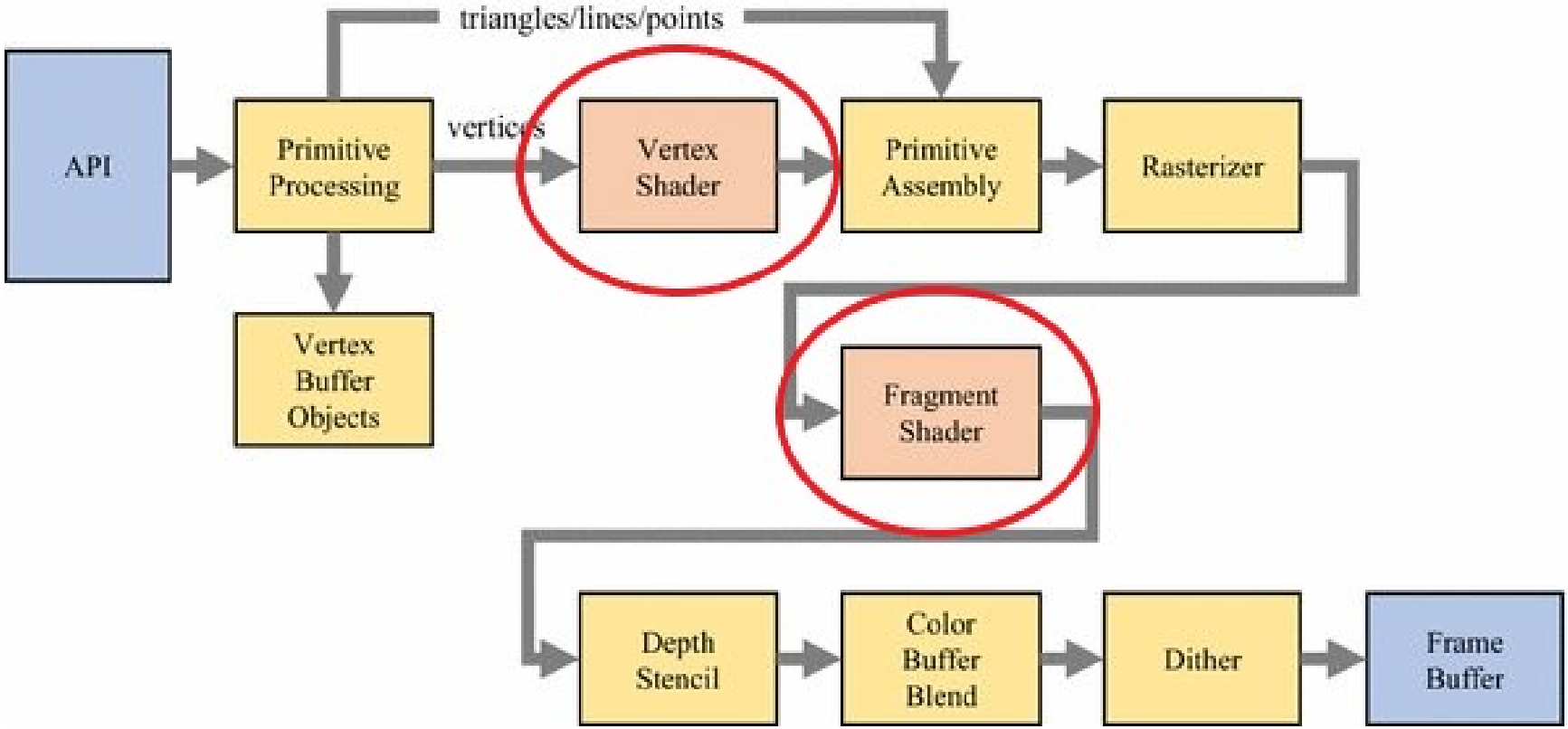
```
// Directive preprocesare
// Shader-e, date (coordonate, culori)
// Proceduri (inițializare, desen)
// Main
    // Inițializări GLUT
    // Generare fereastră
    // Apelare procedură inițializare
        // Creare VBO / VAO
        // Creare / Apelare shader-e
    // Apelare procedură desen
    // Ștergere Shader-e, VBO / VAO
    // Apelare glutMainLoop
```

“Fixed *versus* programmable pipeline”



Sursa: Baek and Kim, 2019

“Fixed *versus* programmable pipeline”



Sursa: Baek and Kim, 2019