1) Desenati un pătrat in loc de triunghi, Colturile avand culorile: rosu, verde, albastru, galben. (Doar de gandit, nu neaparat si de implementat) Ce ar trebui sa facem daca am vrea sa avem centrul patratului negru, de exemplu (si culoarea lui sa fie in degrade spre culorile din varfuri)?

Pași:

1. Vom presupune că desenăm pătratul centrat pe ecrann cu laturile paralele cu laturile ecranului. Modificați coordonatele celor 3 vertecși existenți astfel încât să reprezinte triunghiul format din latura din stânga, latura de sus și diagonala pătratului.
2. Definiți încă 3 vertecși cu culorile cerute în enunț. Schimbați coordonatele
3. Trimiteți 6 puncte în funcția de desenare (parametrul al treilea pentru **glDrawArrays(mod\_desenare, primul\_index\_de\_vertex, nr\_vertecși)**). Atenție, **nu** schimbați numărul din 3 în 6 în funcțiile de trimitere a valorilor (glVertexAttribPointer) spre atribute, deoarece acel 3 se referă la 3 valori float pentru poziție, respectiv culoare, și nu la numărul de vertecși).

2) **Evenimente de mouse.** Pentru a observa mai bine in ce jumatate a ecranului sunteti, desenati o linie alba verticala centrata pe ecran care sa apara peste forma desenată. Pentru a desena linia:

1. în funcția Init():
   1. creați un vector de 2 vertecși (numit lineVerticesData) cu coordonatele liniei astfel încât să se întindă pe verticală pe mijlocul ecranului. Nu mai setați culori pentru vertecși deoarece culoarea liniei e constantă (albă)
   2. veți crea un id de buffer nou (cu glGenBuffers) salvat în variabila lineVboId (de tip GLuint)
   3. folosind funcția glBindBuffer, veți deschide (și asocia) un buffer de tip GL\_ARRAY\_BUFFER cu id-ul lineVboId
   4. cu glBufferData veți seta datele bufferului preluându-le din vectorul lineVerticesData
   5. închideți bufferul cu glBindBuffer(GL\_ARRAY\_BUFFER, 0)
2. Crearea shader-ului liniei:
   1. creați un vertex shader si un fragment shader speciale pentru linie. Puteți porni de la shaderele pentru triunghi, doar că de data asta nu mai interpolați culoarea (setați în fragment shader direct culoarea albă). Veți intra cu Windows Explorer în folderul Resources/Shaders unde creați cele două fișiere. Apoi în Visual Studio, în filtrul Shaders le adăugați cu Add -> Existing Item
   2. în vertex shader nu mai înmulțiti cu MVP pozițiile vertecșilor, deoarece linia trebuie să rămână fixă pe ecran.
   3. creați o variabilă de tip Shaders numită lineShader. Apelați în funcția Init() din NewTrainingFramework.cpp, metoda Init() a obiectului lineShader cu căile către cele două fișiere shader.
3. în funcția Draw() veți adăuga codul pentru desenarea liniei imediat după codul pentru desenarea triunghiului/pătratului (practic după apelul glDrawArrays() existent).
   1. veți folosi shader-ul pentru linie (glUseProgram)
   2. veți deschide bufferul liniei, cu glBindBuffer(), folosind lineVboId
   3. veți transmite datele pentru linie. Veți verifica pentru fiecare variabilă din shader dacă locația e diferită de -1 (în acest caz ar trebui sa fie doar poziția vertecșilor) și veți transmite datele cu glVertexAttribPointer()
   4. veți desena linia cu glDrawArrays() folosind modul de desenare GL\_LINES, și trimițând un număr de 2 vertecși.

Dorim ca atunci când se face click în jumătatea din stânga a ecranului, forma (triunghiul sau pătratul) se va roti spre stânga, iar când faceți click în dreapta, se va roti spre dreapta.

Optional: implementați o mișcare accelerată, dacă se face click spre aceeași direcție în care se rotește deja.

Task-ul se va realiza înregistrând o funcție pentru mouse, asa cum aveți și pentru tastatură (veți face exact aceleași operații realizate pentru funcția Key).

Pentru tipul de buton apăsat si tipul de eveniment veți face câte un enum (de exemplu, *MouseButtons*, și *MouseEvents*). E indicat ca aceste enum-uri sa fie definite într-un header separat, sepcial. Enumul pentru MouseButtons va cuprinde valori pentru butonul drept si stang (puteți adăuga mai târziu si pentru alte butoane precum cel din mijloc, sau cel de scroll). Enumul pentru MouseEvents va cuprinde valori pentru buton apăsat (BTN\_DOWN) și buton eliberat (BTN\_UP), dar și aici puteți adăuga alte valori mai târziu.

În esUtil.h veți defini în clasa ESContext o proprietate callback mouseFunc, asemănătoare cu keyFunc, doar că va avea parametrii: (ESContext\*, *MouseButtons* btn, *MouseEvents event, int x, int y*).

Explicație parametri:

* **esContext** - obiectul context, care are toate informațiile despre aplicație
* **btn** - butonul stang sau drept apasat
* **event** - tipul de eveniment(click, dblclick etc)
* **x, y** - coordonatele la care s-a realizat evenimentul

Veți defini o funcție esRegisterMouseFunc în esUtil.h/esUtil.cpp după modelul funcției esRegisterKeyFunc:

void ESUTIL\_API esRegisterMouseFunc ( ESContext \*esContext,

void (ESCALLBACK \*mouseFunc) (ESContext\*, MouseButtons btn, MouseEvents event, int x, int y) )

care va seta proprietatea mouseFunc.

In esUtil\_win.cpp, în funcția ESWindowProc(care trateaza evenimentul de la utilizator), se vor defini evenimentele: WM\_LBUTTONDOWN, WM\_LBUTTONUP, WM\_RBUTTONDOWN, WM\_RBUTTONUP. Definirea lor se va face după exemplul evenimentelor WM\_KEYDOWN și WM\_KEYUP:

* se preia întâi esContext
* se verifică dacă există și dacă e setată și proprietatea lui, mouseFunc
* se preiau apoi x-ul și y-ul din parametrul lParam. Pentru a afla coordonatele se folosesc [GET\_X\_LPARAM(lParam)](https://learn.microsoft.com/en-us/windows/win32/api/windowsx/nf-windowsx-get_x_lparam), [GET\_Y\_LPARAM(lParam)](https://learn.microsoft.com/en-us/windows/win32/api/windowsx/nf-windowsx-get_y_lparam) Pentru a le putea folosi, trebuie să includeti <windowsx.h>.
* apelați apoi metoda mouseFunc din esContext

În NewTrainingFramework.cpp, așa cum aveți definită funcția

void Key ( ESContext \*esContext, unsigned char key, bool bIsPressed)

veți defini și funcția

void Mouse ( ESContext \*esContext, MouseButtons btn, MouseEvents event, int x, int y)

în care veți trata evenimentele de mouse. Aceasta va trebui înregistrată în \_tmain, așa cum e funcția Key:

esRegisterKeyFunc ( &esContext, Key);

dar folosind esRegisterMouseFunc

**Important!** Pentru cei care aveți nevoie mai târziu în proiect, ca sa mearga dublu-click, trebuie setat in stilul ferestrei si flagul CS\_DBLCLKS (wndclass.style din functia WinCreate (esUtil\_win.cpp).