

Proiect 1 - Grafică pe Calculator
- Depășire între 2 dreptunghiuri -

Linte Robert Ovidiu
Popescu Paullo Robertto Karloss
Grupa 331

Conf. Dr. Stupariu Mihai-Sorin

- 2 Noiembrie 2022 -

Cuprins

1	Introducere	2
1.1	Modul de organizare al echipei	2
1.2	Obiectivele proiectului	2
1.3	Vizionarea proiectului	2
2	Desenarea obiectelor	3
2.1	Prezentarea tablei de joc	3
2.2	Cum a fost construită tabla de joc	4
2.3	Cod sursă	4
3	Adăugarea translațiilor	15
3.1	Prezentarea Translației 1	15
3.1.1	Cod sursă	16
3.2	Prezentarea Translației 2	16
3.2.1	Cod sursă	17
3.3	Prezentarea Translației 3	18
3.3.1	Cod sursă	18
4	Afișarea câștigătorului	19
4.1	Prezentarea înainte de translația finală	19
4.1.1	Cod sursă	19
4.2	Prezentare după translație	19
4.2.1	Cod sursă	20
5	Codul Sursă Complet	21
6	Referințe	39

Introducere

1.1 Modul de organizare al echipei

Echipa noastră:

- Linte Robert Ovidiu
- Popescu Paullo Robertto Karloss

Se va specifica **numele** membrului care a **contribuit** la realizarea fiecărei *etape*.

1.2 Obiectivele proiectului

Simularea unei ”depășiri”:

- O mașină (un dreptunghi) se deplasează pe o șosea uniform (print translație)
- O altă mașină (alt dreptunghi) vine din spate (tot prin translații)
- La un moment dat a doua mașină intră în depășire
- A doua mașină trece în fața primei mașini
- Se afișează la final câștigătorul ”cursei”

Aprofundarea cunoștințelor în OpenGL prin:

- Utilizarea translațiilor
- Desenarea obiectelor
- Utilizarea culorilor

1.3 Vizionarea proiectului

Puteți viziona *demo-ul* proiectului [**aici**](#), iar *repository-ul* de pe Github [**aici**](#).

Desenarea obiectelor

Această etapă a fost realizată de *Popescu Paullo Robertto Karloss*.

2.1 Prezentarea tablei de joc

Tabla de joc conține:

- Șoseua propiu-zisă
- O linie punctată pe post de marcaj rutier
- Două mașini de culori diferite (prima roșie, a doua albastră) care au forma unor dreptunghiuri
- Iarbă pe marginea șoselei
- Un text la finalul șoselei cu mesajul "FINISH", pentru scoate în evidență câștigătorul "cursei"

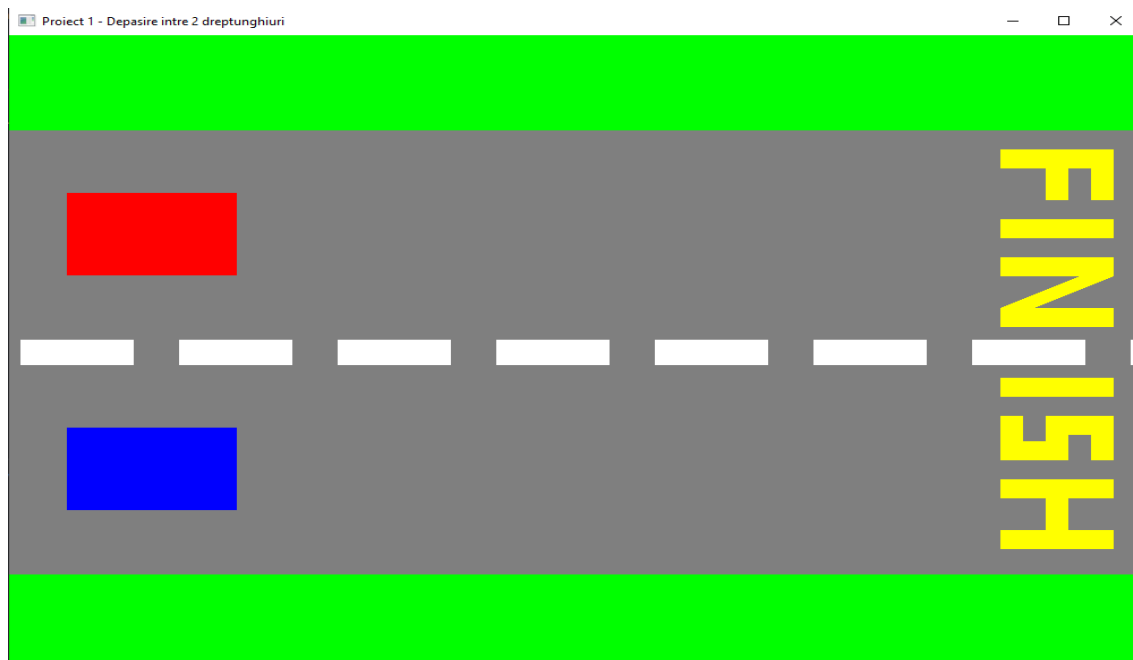


Figura 2.1: Tabla de joc

2.2 Cum a fost construită tabla de joc

Pentru a construi tabla de joc am creat un background (un dreptunghi) de culoare verde. Peste acesta am adăugat șoseaua (un dreptunghi de culoare gri). După care, în interiorul șoselei am creat două dreptunghiuri, unul roșu și unul albastru, pe post de mașini, dreptunghiuri albe pe post de linie punctată (marcaj rutier din legislație) și alte dreptunghiuri de culoare galbenă pentru a scrie mesajul *"FINISH"*.

2.3 Cod sursă

Code 2.1: Crearea punctelor și a culorilor pentru Tabla de joc

```
1 void CreateVBO(void)
2 {
3     // varfurile
4     GLfloat Vertices[] = {
5         // varfuri pentru axe
6         -500.0f, 500.0f, 0.0f, 1.0f,
7         500.0f, 500.0f, 0.0f, 1.0f,
8         500.0f, -500.0f, 0.0f, 1.0f,
9         -500.0f, -500.0f, 0.0f, 1.0f,
10
11         -500.0f, 350.0f, 0.0f, 1.0f,
12         500.0f, 350.0f, 0.0f, 1.0f,
13         500.0f, -350.0f, 0.0f, 1.0f,
14         -500.0f, -350.0f, 0.0f, 1.0f,
15
16         -490.0f, 20.0f, 0.0f, 1.0f,
17         -390.0f, 20.0f, 0.0f, 1.0f,
18         -390.0f, -20.0f, 0.0f, 1.0f,
19         -490.0f, -20.0f, 0.0f, 1.0f,
20
21         -350.0f, 20.0f, 0.0f, 1.0f,
22         -250.0f, 20.0f, 0.0f, 1.0f,
23         -250.0f, -20.0f, 0.0f, 1.0f,
24         -350.0f, -20.0f, 0.0f, 1.0f,
25
26         -210.0f, 20.0f, 0.0f, 1.0f,
27         -110.0f, 20.0f, 0.0f, 1.0f,
28         -110.0f, -20.0f, 0.0f, 1.0f,
29         -210.0f, -20.0f, 0.0f, 1.0f,
```

```

30
31     -70.0f, 20.0f, 0.0f, 1.0f,
32     30.0f, 20.0f, 0.0f, 1.0f,
33     30.0f, -20.0f, 0.0f, 1.0f,
34     -70.0f, -20.0f, 0.0f, 1.0f,
35
36     70.0f, 20.0f, 0.0f, 1.0f,
37     170.0f, 20.0f, 0.0f, 1.0f,
38     170.0f, -20.0f, 0.0f, 1.0f,
39     70.0f, -20.0f, 0.0f, 1.0f,
40
41     210.0f, 20.0f, 0.0f, 1.0f,
42     310.0f, 20.0f, 0.0f, 1.0f,
43     310.0f, -20.0f, 0.0f, 1.0f,
44     210.0f, -20.0f, 0.0f, 1.0f,
45
46     350.0f, 20.0f, 0.0f, 1.0f,
47     450.0f, 20.0f, 0.0f, 1.0f,
48     450.0f, -20.0f, 0.0f, 1.0f,
49     350.0f, -20.0f, 0.0f, 1.0f,
50
51     490.0f, 20.0f, 0.0f, 1.0f,
52     500.0f, 20.0f, 0.0f, 1.0f,
53     500.0f, -20.0f, 0.0f, 1.0f,
54     490.0f, -20.0f, 0.0f, 1.0f,
55
56     -450.0f, 250.0f, 0.0f, 1.0f,
57     -300.0f, 250.0f, 0.0f, 1.0f,
58     -300.0f, 120.0f, 0.0f, 1.0f,
59     -450.0f, 120.0f, 0.0f, 1.0f,
60
61     -450.0f, -250.0f, 0.0f, 1.0f,
62     -300.0f, -250.0f, 0.0f, 1.0f,
63     -300.0f, -120.0f, 0.0f, 1.0f,
64     -450.0f, -120.0f, 0.0f, 1.0f,
65
66     //FINISH
67     //F
68     475.0f, 320.0f, 0.0f, 1.0f,
69     375.0f, 320.0f, 0.0f, 1.0f,
70     375.0f, 290.0f, 0.0f, 1.0f,

```

```

71      475.0f, 290.0f, 0.0f, 1.0f,
72
73      475.0f, 290.0f, 0.0f, 1.0f,
74      475.0f, 240.0f, 0.0f, 1.0f,
75      455.0f, 240.0f, 0.0f, 1.0f,
76      455.0f, 290.0f, 0.0f, 1.0f,
77
78      435.0f, 290.0f, 0.0f, 1.0f,
79      435.0f, 240.0f, 0.0f, 1.0f,
80      415.0f, 240.0f, 0.0f, 1.0f,
81      415.0f, 290.0f, 0.0f, 1.0f,
82      //I
83      475.0f, 210.0f, 0.0f, 1.0f,
84      375.0f, 210.0f, 0.0f, 1.0f,
85      375.0f, 180.0f, 0.0f, 1.0f,
86      475.0f, 180.0f, 0.0f, 1.0f,
87      //N
88      475.0f, 150.0f, 0.0f, 1.0f,
89      375.0f, 150.0f, 0.0f, 1.0f,
90      375.0f, 120.0f, 0.0f, 1.0f,
91      475.0f, 120.0f, 0.0f, 1.0f,
92
93      475.0f, 70.0f, 0.0f, 1.0f,
94      375.0f, 70.0f, 0.0f, 1.0f,
95      375.0f, 40.0f, 0.0f, 1.0f,
96      475.0f, 40.0f, 0.0f, 1.0f,
97
98      475.0f, 120.0f, 0.0f, 1.0f,
99      445.0f, 120.0f, 0.0f, 1.0f,
100     375.0f, 70.0f, 0.0f, 1.0f,
101     405.0f, 70.0f, 0.0f, 1.0f,
102
103     //I
104     475.0f, -40.0f, 0.0f, 1.0f,
105     375.0f, -40.0f, 0.0f, 1.0f,
106     375.0f, -70.0f, 0.0f, 1.0f,
107     475.0f, -70.0f, 0.0f, 1.0f,
108
109     //S
110     475.0f, -100.0f, 0.0f, 1.0f,
111     455.0f, -100.0f, 0.0f, 1.0f,

```

```

112 455.0f, -170.0f, 0.0f, 1.0f,
113 475.0f, -170.0f, 0.0f, 1.0f,
114
115 475.0f, -100.0f, 0.0f, 1.0f,
116 415.0f, -100.0f, 0.0f, 1.0f,
117 415.0f, -130.0f, 0.0f, 1.0f,
118 475.0f, -130.0f, 0.0f, 1.0f,
119
120 415.0f, -100.0f, 0.0f, 1.0f,
121 435.0f, -100.0f, 0.0f, 1.0f,
122 435.0f, -170.0f, 0.0f, 1.0f,
123 415.0f, -170.0f, 0.0f, 1.0f,
124
125 415.0f, -170.0f, 0.0f, 1.0f,
126 415.0f, -140.0f, 0.0f, 1.0f,
127 375.0f, -140.0f, 0.0f, 1.0f,
128 375.0f, -170.0f, 0.0f, 1.0f,
129
130 375.0f, -100.0f, 0.0f, 1.0f,
131 395.0f, -100.0f, 0.0f, 1.0f,
132 395.0f, -170.0f, 0.0f, 1.0f,
133 375.0f, -170.0f, 0.0f, 1.0f,
134
135 //H
136 475.0f, -200.0f, 0.0f, 1.0f,
137 375.0f, -200.0f, 0.0f, 1.0f,
138 375.0f, -230.0f, 0.0f, 1.0f,
139 475.0f, -230.0f, 0.0f, 1.0f,
140
141 475.0f, -280.0f, 0.0f, 1.0f,
142 375.0f, -280.0f, 0.0f, 1.0f,
143 375.0f, -310.0f, 0.0f, 1.0f,
144 475.0f, -310.0f, 0.0f, 1.0f,
145
146 435.0f, -310.0f, 0.0f, 1.0f,
147 415.0f, -310.0f, 0.0f, 1.0f,
148 415.0f, -200.0f, 0.0f, 1.0f,
149 435.0f, -200.0f, 0.0f, 1.0f,
150
151 //Blue win
152

```



```

153 //B
154 -300.0f, 700.0f, 0.0f, 1.0f,
155 -300.0f, 600.0f, 0.0f, 1.0f,
156 -280.0f, 600.f, 0.0f, 1.0f,
157 -280.0f, 700.0f, 0.0f, 1.0f,
158
159 -250.0f, 700.0f, 0.0f, 1.0f,
160 -250.0f, 600.0f, 0.0f, 1.0f,
161 -230.0f, 600.f, 0.0f, 1.0f,
162 -230.0f, 700.0f, 0.0f, 1.0f,
163
164 -300.0f, 700.0f, 0.0f, 1.0f,
165 -300.0f, 680.0f, 0.0f, 1.0f,
166 -230.0f, 680.f, 0.0f, 1.0f,
167 -230.0f, 700.0f, 0.0f, 1.0f,
168
169 -300.0f, 660.0f, 0.0f, 1.0f,
170 -300.0f, 640.0f, 0.0f, 1.0f,
171 -230.0f, 640.f, 0.0f, 1.0f,
172 -230.0f, 660.0f, 0.0f, 1.0f,
173
174 -300.0f, 620.0f, 0.0f, 1.0f,
175 -300.0f, 600.0f, 0.0f, 1.0f,
176 -230.0f, 600.f, 0.0f, 1.0f,
177 -230.0f, 620.0f, 0.0f, 1.0f,
178
179 //L
180 -200.0f, 700.0f, 0.0f, 1.0f,
181 -200.0f, 600.0f, 0.0f, 1.0f,
182 -180.0f, 600.f, 0.0f, 1.0f,
183 -180.0f, 700.0f, 0.0f, 1.0f,
184
185 -200.0f, 620.0f, 0.0f, 1.0f,
186 -200.0f, 600.0f, 0.0f, 1.0f,
187 -150.0f, 600.f, 0.0f, 1.0f,
188 -150.0f, 620.0f, 0.0f, 1.0f,
189
190 //U
191 -120.0f, 700.0f, 0.0f, 1.0f,
192 -120.0f, 600.0f, 0.0f, 1.0f,
193 -100.0f, 600.f, 0.0f, 1.0f,

```

```

194     -100.0f, 700.0f, 0.0f, 1.0f,
195
196     -70.0f, 700.0f, 0.0f, 1.0f,
197     -70.0f, 600.0f, 0.0f, 1.0f,
198     -50.0f, 600.f, 0.0f, 1.0f,
199     -50.0f, 700.0f, 0.0f, 1.0f,
200
201     -120.0f, 620.0f, 0.0f, 1.0f,
202     -50.0f, 620.0f, 0.0f, 1.0f,
203     -50.0f, 600.0f, 0.0f, 1.0f,
204     -120.0f, 600.0f, 0.0f, 1.0f,
205
206     //E
207     -20.0f, 700.0f, 0.0f, 1.0f,
208     -20.0f, 600.0f, 0.0f, 1.0f,
209     0.0f, 600.f, 0.0f, 1.0f,
210     0.0f, 700.0f, 0.0f, 1.0f,
211
212     -20.0f, 620.0f, 0.0f, 1.0f,
213     30.0f, 620.0f, 0.0f, 1.0f,
214     30.0f, 600.0f, 0.0f, 1.0f,
215     -20.0f, 600.0f, 0.0f, 1.0f,
216
217     -20.0f, 660.0f, 0.0f, 1.0f,
218     30.0f, 660.0f, 0.0f, 1.0f,
219     30.0f, 640.0f, 0.0f, 1.0f,
220     -20.0f, 640.0f, 0.0f, 1.0f,
221
222     -20.0f, 700.0f, 0.0f, 1.0f,
223     30.0f, 700.0f, 0.0f, 1.0f,
224     30.0f, 680.0f, 0.0f, 1.0f,
225     -20.0f, 680.0f, 0.0f, 1.0f,
226
227     //W
228     100.0f, 700.0f, 0.0f, 1.0f,
229     120.0f, 700.0f, 0.0f, 1.0f,
230     140.0f, 600.0f, 0.0f, 1.0f,
231     120.0f, 600.0f, 0.0f, 1.0f,
232
233     140.0f, 600.0f, 0.0f, 1.0f,
234     120.0f, 600.0f, 0.0f, 1.0f,

```

```

235     140.0f, 670.0f, 0.0f, 1.0f,
236     160.0f, 670.0f, 0.0f, 1.0f,
237
238     140.0f, 670.0f, 0.0f, 1.0f,
239     160.0f, 670.0f, 0.0f, 1.0f,
240     180.0f, 600.0f, 0.0f, 1.0f,
241     160.0f, 600.0f, 0.0f, 1.0f,
242
243     180.0f, 600.0f, 0.0f, 1.0f,
244     160.0f, 600.0f, 0.0f, 1.0f,
245     180.0f, 700.0f, 0.0f, 1.0f,
246     200.0f, 700.0f, 0.0f, 1.0f,
247
248     //I
249
250     220.0f, 700.0f, 0.0f, 1.0f,
251     240.0f, 700.0f, 0.0f, 1.0f,
252     240.0f, 600.0f, 0.0f, 1.0f,
253     220.0f, 600.0f, 0.0f, 1.0f,
254
255     //N
256
257     260.0f, 700.0f, 0.0f, 1.0f,
258     280.0f, 700.0f, 0.0f, 1.0f,
259     280.0f, 600.0f, 0.0f, 1.0f,
260     260.0f, 600.0f, 0.0f, 1.0f,
261
262     310.0f, 700.0f, 0.0f, 1.0f,
263     330.0f, 700.0f, 0.0f, 1.0f,
264     330.0f, 600.0f, 0.0f, 1.0f,
265     310.0f, 600.0f, 0.0f, 1.0f,
266
267     280.0f, 700.0f, 0.0f, 1.0f,
268     280.0f, 670.0f, 0.0f, 1.0f,
269     310.0f, 600.0f, 0.0f, 1.0f,
270     310.0f, 630.0f, 0.0f, 1.0f,
271 };
272
273 // culorile varfurilor din colturi
274 GLfloat Colors[] = {
275     0.0f, 1.0f, 0.0f, 1.0f,

```

276	0.0f, 1.0f, 0.0f, 1.0f,
277	0.0f, 1.0f, 0.0f, 1.0f,
278	0.0f, 1.0f, 0.0f, 1.0f,
279	
280	0.5f, 0.5f, 0.5f, 1.0f,
281	0.5f, 0.5f, 0.5f, 1.0f,
282	0.5f, 0.5f, 0.5f, 1.0f,
283	0.5f, 0.5f, 0.5f, 1.0f,
284	
285	1.0f, 1.0f, 1.0f, 1.0f,
286	1.0f, 1.0f, 1.0f, 1.0f,
287	1.0f, 1.0f, 1.0f, 1.0f,
288	1.0f, 1.0f, 1.0f, 1.0f,
289	
290	1.0f, 1.0f, 1.0f, 1.0f,
291	1.0f, 1.0f, 1.0f, 1.0f,
292	1.0f, 1.0f, 1.0f, 1.0f,
293	1.0f, 1.0f, 1.0f, 1.0f,
294	
295	1.0f, 1.0f, 1.0f, 1.0f,
296	1.0f, 1.0f, 1.0f, 1.0f,
297	1.0f, 1.0f, 1.0f, 1.0f,
298	1.0f, 1.0f, 1.0f, 1.0f,
299	
300	1.0f, 1.0f, 1.0f, 1.0f,
301	1.0f, 1.0f, 1.0f, 1.0f,
302	1.0f, 1.0f, 1.0f, 1.0f,
303	1.0f, 1.0f, 1.0f, 1.0f,
304	
305	1.0f, 1.0f, 1.0f, 1.0f,
306	1.0f, 1.0f, 1.0f, 1.0f,
307	1.0f, 1.0f, 1.0f, 1.0f,
308	1.0f, 1.0f, 1.0f, 1.0f,
309	
310	1.0f, 1.0f, 1.0f, 1.0f,
311	1.0f, 1.0f, 1.0f, 1.0f,
312	1.0f, 1.0f, 1.0f, 1.0f,
313	1.0f, 1.0f, 1.0f, 1.0f,
314	
315	1.0f, 1.0f, 1.0f, 1.0f,
316	1.0f, 1.0f, 1.0f, 1.0f,

317	1.0f, 1.0f, 1.0f, 1.0f,
318	1.0f, 1.0f, 1.0f, 1.0f,
319	
320	1.0f, 1.0f, 1.0f, 1.0f,
321	1.0f, 1.0f, 1.0f, 1.0f,
322	1.0f, 1.0f, 1.0f, 1.0f,
323	1.0f, 1.0f, 1.0f, 1.0f,
324	
325	1.0f, 0.0f, 0.0f, 1.0f,
326	1.0f, 0.0f, 0.0f, 1.0f,
327	1.0f, 0.0f, 0.0f, 1.0f,
328	1.0f, 0.0f, 0.0f, 1.0f,
329	
330	0.0f, 0.0f, 1.0f, 1.0f,
331	0.0f, 0.0f, 1.0f, 1.0f,
332	0.0f, 0.0f, 1.0f, 1.0f,
333	0.0f, 0.0f, 1.0f, 1.0f,
334	
335	1.0f, 1.0f, 0.0f, 1.0f,
336	1.0f, 1.0f, 0.0f, 1.0f,
337	1.0f, 1.0f, 0.0f, 1.0f,
338	1.0f, 1.0f, 0.0f, 1.0f,
339	
340	1.0f, 1.0f, 0.0f, 1.0f,
341	1.0f, 1.0f, 0.0f, 1.0f,
342	1.0f, 1.0f, 0.0f, 1.0f,
343	1.0f, 1.0f, 0.0f, 1.0f,
344	
345	1.0f, 1.0f, 0.0f, 1.0f,
346	1.0f, 1.0f, 0.0f, 1.0f,
347	1.0f, 1.0f, 0.0f, 1.0f,
348	1.0f, 1.0f, 0.0f, 1.0f,
349	
350	1.0f, 1.0f, 0.0f, 1.0f,
351	1.0f, 1.0f, 0.0f, 1.0f,
352	1.0f, 1.0f, 0.0f, 1.0f,
353	1.0f, 1.0f, 0.0f, 1.0f,
354	
355	1.0f, 1.0f, 0.0f, 1.0f,
356	1.0f, 1.0f, 0.0f, 1.0f,
357	1.0f, 1.0f, 0.0f, 1.0f,

358	1.0f, 1.0f, 0.0f, 1.0f,
359	
360	1.0f, 1.0f, 0.0f, 1.0f,
361	1.0f, 1.0f, 0.0f, 1.0f,
362	1.0f, 1.0f, 0.0f, 1.0f,
363	1.0f, 1.0f, 0.0f, 1.0f,
364	
365	1.0f, 1.0f, 0.0f, 1.0f,
366	1.0f, 1.0f, 0.0f, 1.0f,
367	1.0f, 1.0f, 0.0f, 1.0f,
368	1.0f, 1.0f, 0.0f, 1.0f,
369	
370	1.0f, 1.0f, 0.0f, 1.0f,
371	1.0f, 1.0f, 0.0f, 1.0f,
372	1.0f, 1.0f, 0.0f, 1.0f,
373	1.0f, 1.0f, 0.0f, 1.0f,
374	
375	1.0f, 1.0f, 0.0f, 1.0f,
376	1.0f, 1.0f, 0.0f, 1.0f,
377	1.0f, 1.0f, 0.0f, 1.0f,
378	1.0f, 1.0f, 0.0f, 1.0f,
379	
380	1.0f, 1.0f, 0.0f, 1.0f,
381	1.0f, 1.0f, 0.0f, 1.0f,
382	1.0f, 1.0f, 0.0f, 1.0f,
383	1.0f, 1.0f, 0.0f, 1.0f,
384	
385	1.0f, 1.0f, 0.0f, 1.0f,
386	1.0f, 1.0f, 0.0f, 1.0f,
387	1.0f, 1.0f, 0.0f, 1.0f,
388	1.0f, 1.0f, 0.0f, 1.0f,
389	
390	1.0f, 1.0f, 0.0f, 1.0f,
391	1.0f, 1.0f, 0.0f, 1.0f,
392	1.0f, 1.0f, 0.0f, 1.0f,
393	1.0f, 1.0f, 0.0f, 1.0f,
394	
395	1.0f, 1.0f, 0.0f, 1.0f,
396	1.0f, 1.0f, 0.0f, 1.0f,
397	1.0f, 1.0f, 0.0f, 1.0f,
398	1.0f, 1.0f, 0.0f, 1.0f,

```

399
400     1.0f, 1.0f, 0.0f, 1.0f,
401     1.0f, 1.0f, 0.0f, 1.0f,
402     1.0f, 1.0f, 0.0f, 1.0f,
403     1.0f, 1.0f, 0.0f, 1.0f,
404
405     1.0f, 1.0f, 0.0f, 1.0f,
406     1.0f, 1.0f, 0.0f, 1.0f,
407     1.0f, 1.0f, 0.0f, 1.0f,
408     1.0f, 1.0f, 0.0f, 1.0f,
409
410     1.0f, 1.0f, 0.0f, 1.0f,
411     1.0f, 1.0f, 0.0f, 1.0f,
412     1.0f, 1.0f, 0.0f, 1.0f,
413     1.0f, 1.0f, 0.0f, 1.0f,
414 };
```

```

606     glDrawArrays(GL_POLYGON, 0, 4);
607     glDrawArrays(GL_POLYGON, 4, 4);
608     glDrawArrays(GL_POLYGON, 8, 4);
609     glDrawArrays(GL_POLYGON, 12, 4);
610     glDrawArrays(GL_POLYGON, 16, 4);
611     glDrawArrays(GL_POLYGON, 20, 4);
612     glDrawArrays(GL_POLYGON, 24, 4);
613     glDrawArrays(GL_POLYGON, 28, 4);
614     glDrawArrays(GL_POLYGON, 32, 4);
615     glDrawArrays(GL_POLYGON, 36, 4);
616     glDrawArrays(GL_POLYGON, 40, 4);
617     glDrawArrays(GL_POLYGON, 44, 4);
618     glDrawArrays(GL_POLYGON, 48, 4);
619     glDrawArrays(GL_POLYGON, 52, 4);
620     glDrawArrays(GL_POLYGON, 56, 4);
621     glDrawArrays(GL_POLYGON, 60, 4);
622     glDrawArrays(GL_POLYGON, 64, 4);
623     glDrawArrays(GL_POLYGON, 68, 4);
624     glDrawArrays(GL_POLYGON, 72, 4);
625     glDrawArrays(GL_POLYGON, 76, 4);
626     glDrawArrays(GL_POLYGON, 80, 4);
627     glDrawArrays(GL_POLYGON, 84, 4);
628     glDrawArrays(GL_POLYGON, 88, 4);
629     glDrawArrays(GL_POLYGON, 92, 4);
630     glDrawArrays(GL_POLYGON, 96, 4);
631     glDrawArrays(GL_POLYGON, 100, 4);
632     glDrawArrays(GL_POLYGON, 104, 4);
633     glDrawArrays(GL_POLYGON, 108, 4);
```

Figura 2.2: Cod OpenGL pentru Tabla de joc

Adăugarea translațiilor

Această etapă a fost realizată de *Linte Robert Ovidiu*.

3.1 Prezentarea Translației 1

Dreptunghiul roșu pleacă cu o viteză inițială mai mică decât a dreptunghiului albastru.

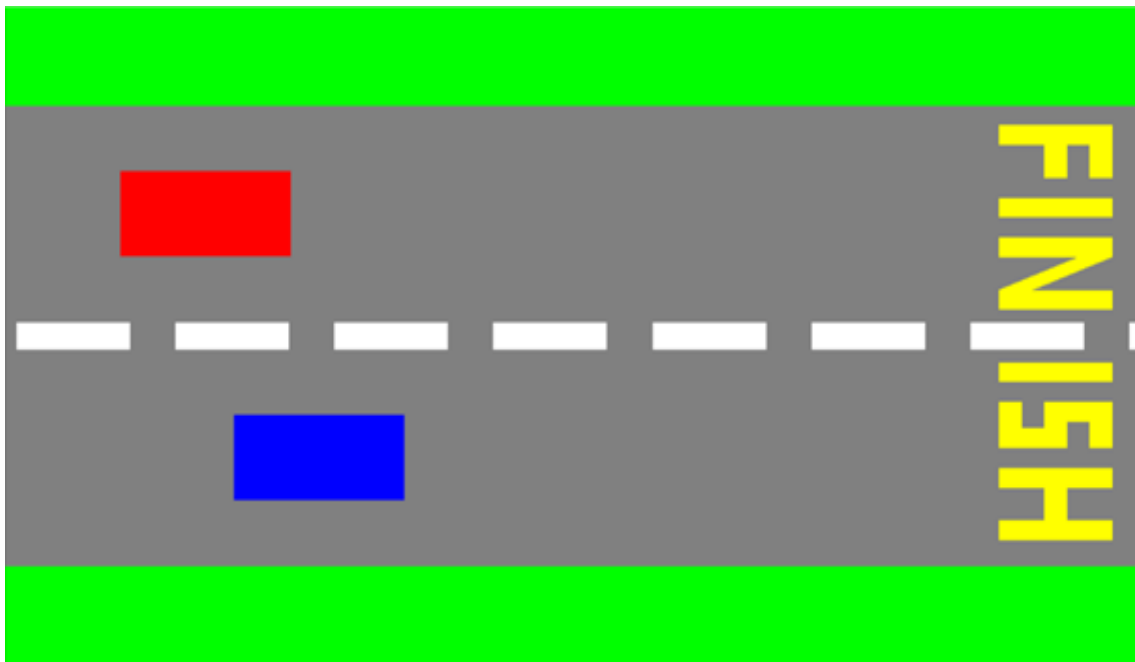


Figura 3.1.1: Deplasarea inițială a dreptunghiurilor

3.1.1 Cod sursă

```
635 // Matricea pentru dreptunghiul rosu
636 myMatrix = resizeMatrix * matrTransl * matrDepl * matrScale2;
637 // Culoarea
638 codCol = 2;
639 // Transmitere variabile uniforme
640 glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);
641 glUniform1i(codColLocation, codCol);
642 // Apelare DrawArrays
643 glDrawArrays(GL_POLYGON, 40, 4);
644
645 // Matricea pentru dreptunghiul rosu
646 myMatrix = resizeMatrix * matrTransl2 * matrDepl * matrScale2 * matrRot;
647 // Culoarea
648 codCol = 1;
649 // Transmitere variabile uniforme
650 glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0]);
651 glUniform1i(codColLocation, codCol);
652 // Apelare DrawArrays
653 glDrawArrays(GL_POLYGON, 44, 4);
654
```

Figura 3.1.2: Cod OpenGL pentru prima translație

```
590 resizeMatrix = glm::ortho(-width, width, -height, height); // scalam, "aducem" scena la "patratul standard" [-1,1]x[-1,1]
591 matrTransl = glm::translate(glm::mat4(1.0f), glm::vec3(i, k, 0.0)); // controleaza translatia de-a lungul lui Ox
592 matrTransl3 = glm::translate(glm::mat4(1.0f), glm::vec3(0.0, h, 0.0));
593 matrDepl = glm::translate(glm::mat4(1.0f), glm::vec3(1.0, 1.0, 0.0)); // plaseaza patratul rosu
594 matrScale2 = glm::scale(glm::mat4(1.0f), glm::vec3(1.0, 1.0, 0.0)); // folosita la desenarea patratului rosu
595 matrTransl2 = glm::translate(glm::mat4(1.0f), glm::vec3(j, l, 0.0));
596 matrRot = glm::rotate(glm::mat4(1.0f), angle, glm::vec3(0.0, 0.0, 1.0)); // rotatie folosita la deplasarea patratului rosu
```

Figura 3.1.3: Cod OpenGL pentru prima translație

```
53 void miscad(void)
54 {
55     if (i > -1 && i <= 400 && j <= 750)
56     {
57         i = i + alpha;
58         alpha = +step;
59     }
60     if (j > -1.0 && j <= 150) {
61         j = j + alpha2;
62         alpha2 = +step2;
63     }
}
```

Figura 3.1.4: Cod OpenGL pentru prima translație

3.2 Prezentarea Translației 2

În momentul în care dreptunghiul albastru, îl depășește total pe cel roșu, începe procesul de depășire (se aplică o rotație de 0.5 pe dreptunghiul albastru cât și o translație pe diagonală, iar în final se aplică o rotație inversă pentru a-l aduce pe poziția inițială, i.e. paralel cu axa Ox).

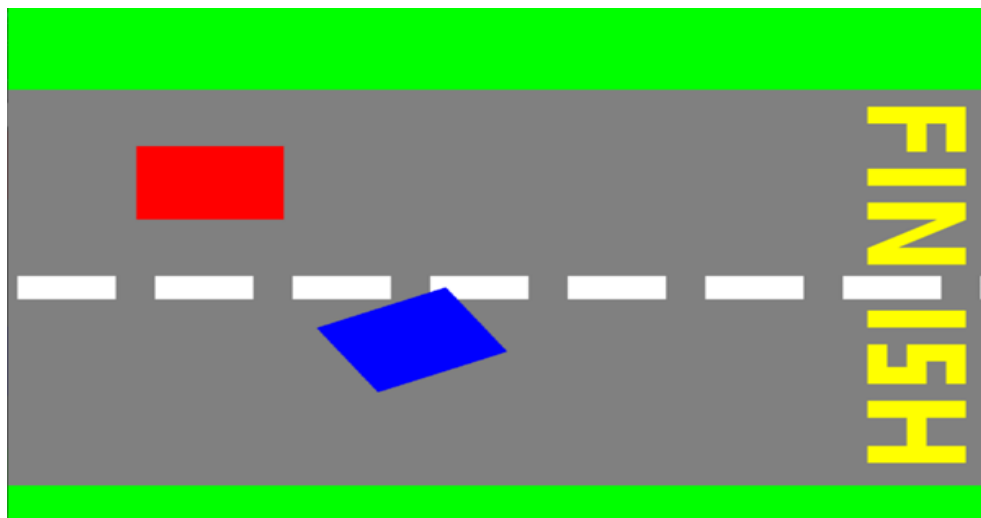


Figura 3.2.1: Depășire dreptunghi roșu

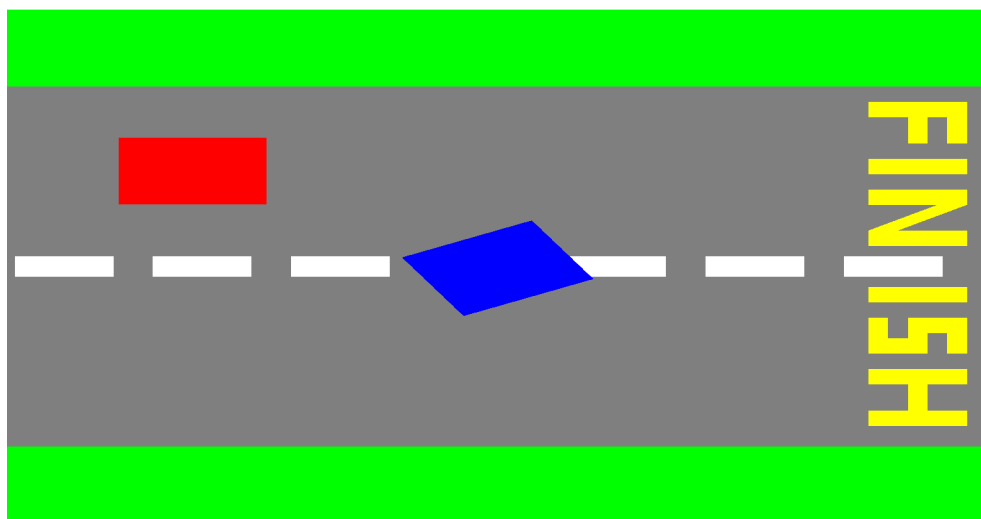


Figura 3.2.2: Depășire dreptunghi roșu

3.2.1 Cod sursă

```

53 void miscad(void)
54 {
55     if (i > -1 && i <= 400 && j <= 750)
56     {
57         i = i + alpha;
58         alpha = +step;
59     }
60     if (j > -1.0 && j <= 150) {
61         j = j + alpha2;
62         alpha2 = +step2;
63     }

```

Figura 3.2.3: Cod OpenGL pentru a doua translație

3.3 Prezentarea Translației 3

După translația 2, dreptunghiul albastru accelerează până trece linia de "FINISH".

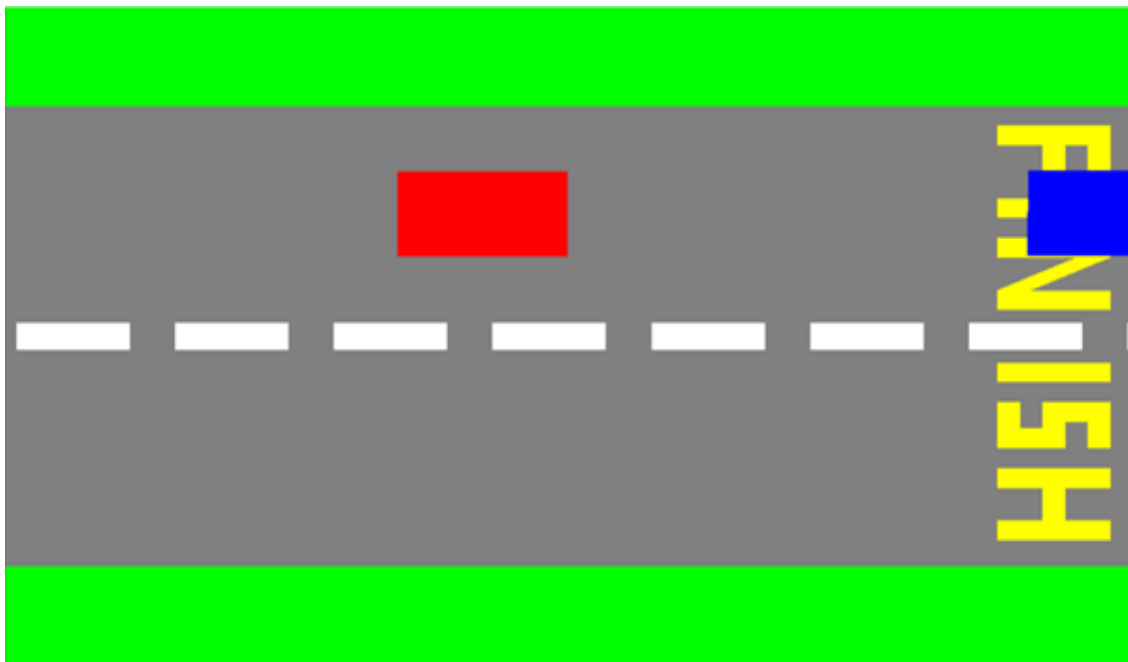


Figura 3.2.1: Accelerare dreptunghi albastru

3.3.1 Cod sursă

```
82  if (l >= 370 && j <= 850 && angle <= 0) {  
83      j = j + alpha2;  
84      alpha2 = +step3;  
85  }
```

Figura 3.3.1: Cod OpenGL pentru a treia translație

Afișarea câștigătorului

4.1 Prezentarea înainte de translația finală

Literele sunt inițial puse în afara ecranului (*nu sunt vizibile în tabla de joc*).

4.1.1 Cod sursă

```
82     if (l >= 370 && j <= 850 && angle <= 0) {  
83         j = j + alpha2;  
84         alpha2 = +step3;  
85     }
```

Figura 4.1.1: Cod OpenGL pentru literele ascunse

Această etapă a fost realizată de *Popescu Paullo Robertto Karloss*.

4.2 Prezentare după translație

După ce dreptunghiul albastru reușește să treacă linia de "FINISH", sunt translatate literele în zona de sus a tablei de joc.

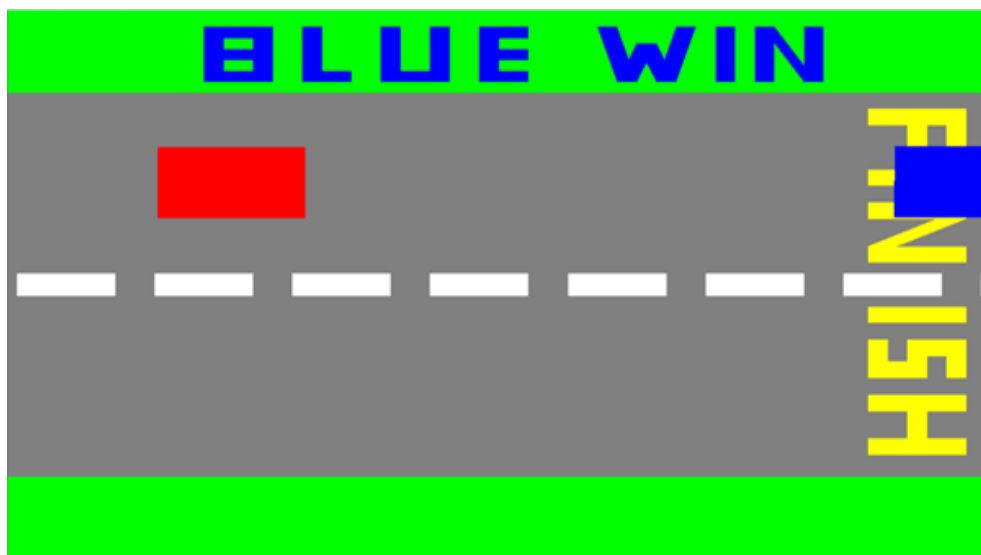


Figura 4.1.1: Cod OpenGL pentru a prima translație

4.2.1 Cod sursă

```
86  if (j >= 750 && h >= -230) {  
87      h = h - alpha4;  
88      alpha4 = +step3;  
89  }
```

Figura 4.1.1: Cod OpenGL pentru translația finală

Această etapă a fost realizată de ***Linte Robert Ovidiu.***

Codul Sursă Complet

Codul îl puteți găsi în fișierul **proiectGrafica.cpp** sau atașat mai jos.

```
1  /* DESCRIERE: DEPASIRE INTRE DREPTUNGHIURI -- varianta cu OpenGL "
   nou"
2  - utilizeaza diverse transformari si compunerea acestora folosind
   biblioteca glm
3  - functii pentru utilizarea mouse-ului
4  */
5
6  #include <windows.h> // biblioteci care urmeaza sa fie incluse
7  #include <stdlib.h> // necesare pentru citirea shader-elor
8  #include <stdio.h>
9  #include <math.h>
10 #include <iostream>
11 #include <GL/glew.h> // glew apare inainte de freeglut
12 #include <GL/freeglut.h> // nu trebuie uitat freeglut.h
13 #include "loadShaders.h"
14
15 // Din biblioteca glm
16 #include "glm/glm.hpp"
17 #include "glm/gtc/matrix_transform.hpp"
18 #include "glm/gtx/transform.hpp"
19 #include "glm/gtc/type_ptr.hpp"
20
21 using namespace std;
22
23 GLuint
24 VaoId,
25 VboId,
26 ColorBufferId,
27 ProgramId,
28 myMatrixLocation,
29 matrScaleLocation,
30 matrTranslLocation,
31 matrRotlLocation,
32 codColLocation;
```

```

33
34 int codCol;
35 float PI = 3.141592, angle = 0;
36 float tx = 0; float ty = 0;
37 float width = 500, height = 500;
38 float i = 0.0, j = 0.0, h = 0.0, k = 0.0, l = 0.0, alpha4 = 0.0,
    alpha2 = 0.0, alpha3 = 0.0, step2 = 0.5, step3 = 1.5, alpha =
    0.0, step = 0.1, beta = 0.003, ok = 0;
39 glm::mat4
40 myMatrix, resizeMatrix, matrTransl, matrTransl2, matrTransl3,
    matrScale1, matrScale2, matrRot, matrDepl;
41
42 void displayMatrix()
43 {
44     for (int ii = 0; ii < 4; ii++)
45     {
46         for (int jj = 0; jj < 4; jj++)
47             cout << myMatrix[ii][jj] << " ";
48         cout << endl;
49     };
50     cout << "\n";
51 };
52
53 void miscad(void)
54 {
55     if (i > -1 && i <= 400 && j <= 750)
56     {
57         i = i + alpha;
58         alpha = +step;
59     }
60     if (j > -1.0 && j <= 150) {
61         j = j + alpha2;
62         alpha2 = +step2;
63     }
64
65     if (j >= 150 && j <= 250 && angle <= 0.5) {
66         angle += beta;
67         l = l + alpha3;
68         alpha3 = +step3;
69     }
70

```

```

71     if (l >= 200 && l <= 370 && angle >= 0.5) {
72         j = j + alpha2;
73         alpha2 = +step2;
74         l = l + alpha3;
75         alpha3 = +step2;
76     }
77     if (l >= 370 && angle <= 0.6 && angle >= 0) {
78         angle -= beta;
79         j = j + alpha2;
80         alpha2 = +step3;
81     }
82     if (l >= 370 && j <= 850 && angle <= 0) {
83         j = j + alpha2;
84         alpha2 = +step3;
85     }
86     if (j >= 750 && h >= -230) {
87         h = h - alpha4;
88         alpha4 = +step3;
89     }
90
91     glutPostRedisplay();
92 }
93
94
95
96
97 void mouse(int button, int state, int x, int y)
98 {
99     switch (button) {
100     case GLUT_LEFT_BUTTON:
101         if (state == GLUT_DOWN) {
102             alpha = -step;
103             alpha2 = -step2;
104         }
105         glutIdleFunc(miscad);
106         break;
107     case GLUT_RIGHT_BUTTON:
108         if (state == GLUT_DOWN) {
109             alpha = step;
110             alpha2 = step2;
111         }

```



```

112     glutIdleFunc(miscad);
113     break;
114     default:
115         break;
116 }
117 }
118
119 void CreateVBO(void)
120 {
121     // varfurile
122     GLfloat Vertices[] = {
123         // varfuri pentru axe
124         -500.0f, 500.0f, 0.0f, 1.0f,
125         500.0f, 500.0f, 0.0f, 1.0f,
126         500.0f, -500.0f, 0.0f, 1.0f,
127         -500.0f, -500.0f, 0.0f, 1.0f,
128
129         -500.0f, 350.0f, 0.0f, 1.0f,
130         500.0f, 350.0f, 0.0f, 1.0f,
131         500.0f, -350.0f, 0.0f, 1.0f,
132         -500.0f, -350.0f, 0.0f, 1.0f,
133
134         -490.0f, 20.0f, 0.0f, 1.0f,
135         -390.0f, 20.0f, 0.0f, 1.0f,
136         -390.0f, -20.0f, 0.0f, 1.0f,
137         -490.0f, -20.0f, 0.0f, 1.0f,
138
139         -350.0f, 20.0f, 0.0f, 1.0f,
140         -250.0f, 20.0f, 0.0f, 1.0f,
141         -250.0f, -20.0f, 0.0f, 1.0f,
142         -350.0f, -20.0f, 0.0f, 1.0f,
143
144         -210.0f, 20.0f, 0.0f, 1.0f,
145         -110.0f, 20.0f, 0.0f, 1.0f,
146         -110.0f, -20.0f, 0.0f, 1.0f,
147         -210.0f, -20.0f, 0.0f, 1.0f,
148
149         -70.0f, 20.0f, 0.0f, 1.0f,
150         30.0f, 20.0f, 0.0f, 1.0f,
151         30.0f, -20.0f, 0.0f, 1.0f,
152         -70.0f, -20.0f, 0.0f, 1.0f,

```

```

153
154 70.0f, 20.0f, 0.0f, 1.0f,
155 170.0f, 20.0f, 0.0f, 1.0f,
156 170.0f, -20.0f, 0.0f, 1.0f,
157 70.0f, -20.0f, 0.0f, 1.0f,
158
159 210.0f, 20.0f, 0.0f, 1.0f,
160 310.0f, 20.0f, 0.0f, 1.0f,
161 310.0f, -20.0f, 0.0f, 1.0f,
162 210.0f, -20.0f, 0.0f, 1.0f,
163
164 350.0f, 20.0f, 0.0f, 1.0f,
165 450.0f, 20.0f, 0.0f, 1.0f,
166 450.0f, -20.0f, 0.0f, 1.0f,
167 350.0f, -20.0f, 0.0f, 1.0f,
168
169 490.0f, 20.0f, 0.0f, 1.0f,
170 500.0f, 20.0f, 0.0f, 1.0f,
171 500.0f, -20.0f, 0.0f, 1.0f,
172 490.0f, -20.0f, 0.0f, 1.0f,
173
174 -450.0f, 250.0f, 0.0f, 1.0f,
175 -300.0f, 250.0f, 0.0f, 1.0f,
176 -300.0f, 120.0f, 0.0f, 1.0f,
177 -450.0f, 120.0f, 0.0f, 1.0f,
178
179 -450.0f, -250.0f, 0.0f, 1.0f,
180 -300.0f, -250.0f, 0.0f, 1.0f,
181 -300.0f, -120.0f, 0.0f, 1.0f,
182 -450.0f, -120.0f, 0.0f, 1.0f,
183
184 //FINISH
185 //F
186 475.0f, 320.0f, 0.0f, 1.0f,
187 375.0f, 320.0f, 0.0f, 1.0f,
188 375.0f, 290.0f, 0.0f, 1.0f,
189 475.0f, 290.0f, 0.0f, 1.0f,
190
191 475.0f, 290.0f, 0.0f, 1.0f,
192 475.0f, 240.0f, 0.0f, 1.0f,
193 455.0f, 240.0f, 0.0f, 1.0f,

```

```

194 455.0f, 290.0f, 0.0f, 1.0f,
195
196 435.0f, 290.0f, 0.0f, 1.0f,
197 435.0f, 240.0f, 0.0f, 1.0f,
198 415.0f, 240.0f, 0.0f, 1.0f,
199 415.0f, 290.0f, 0.0f, 1.0f,
200 //I
201 475.0f, 210.0f, 0.0f, 1.0f,
202 375.0f, 210.0f, 0.0f, 1.0f,
203 375.0f, 180.0f, 0.0f, 1.0f,
204 475.0f, 180.0f, 0.0f, 1.0f,
205 //N
206 475.0f, 150.0f, 0.0f, 1.0f,
207 375.0f, 150.0f, 0.0f, 1.0f,
208 375.0f, 120.0f, 0.0f, 1.0f,
209 475.0f, 120.0f, 0.0f, 1.0f,
210
211 475.0f, 70.0f, 0.0f, 1.0f,
212 375.0f, 70.0f, 0.0f, 1.0f,
213 375.0f, 40.0f, 0.0f, 1.0f,
214 475.0f, 40.0f, 0.0f, 1.0f,
215
216 475.0f, 120.0f, 0.0f, 1.0f,
217 445.0f, 120.0f, 0.0f, 1.0f,
218 375.0f, 70.0f, 0.0f, 1.0f,
219 405.0f, 70.0f, 0.0f, 1.0f,
220
221 //I
222 475.0f, -40.0f, 0.0f, 1.0f,
223 375.0f, -40.0f, 0.0f, 1.0f,
224 375.0f, -70.0f, 0.0f, 1.0f,
225 475.0f, -70.0f, 0.0f, 1.0f,
226
227 //S
228 475.0f, -100.0f, 0.0f, 1.0f,
229 455.0f, -100.0f, 0.0f, 1.0f,
230 455.0f, -170.0f, 0.0f, 1.0f,
231 475.0f, -170.0f, 0.0f, 1.0f,
232
233 475.0f, -100.0f, 0.0f, 1.0f,
234 415.0f, -100.0f, 0.0f, 1.0f,

```

```

235 415.0f, -130.0f, 0.0f, 1.0f,
236 475.0f, -130.0f, 0.0f, 1.0f,
237
238 415.0f, -100.0f, 0.0f, 1.0f,
239 435.0f, -100.0f, 0.0f, 1.0f,
240 435.0f, -170.0f, 0.0f, 1.0f,
241 415.0f, -170.0f, 0.0f, 1.0f,
242
243 415.0f, -170.0f, 0.0f, 1.0f,
244 415.0f, -140.0f, 0.0f, 1.0f,
245 375.0f, -140.0f, 0.0f, 1.0f,
246 375.0f, -170.0f, 0.0f, 1.0f,
247
248 375.0f, -100.0f, 0.0f, 1.0f,
249 395.0f, -100.0f, 0.0f, 1.0f,
250 395.0f, -170.0f, 0.0f, 1.0f,
251 375.0f, -170.0f, 0.0f, 1.0f,
252
253 //H
254 475.0f, -200.0f, 0.0f, 1.0f,
255 375.0f, -200.0f, 0.0f, 1.0f,
256 375.0f, -230.0f, 0.0f, 1.0f,
257 475.0f, -230.0f, 0.0f, 1.0f,
258
259 475.0f, -280.0f, 0.0f, 1.0f,
260 375.0f, -280.0f, 0.0f, 1.0f,
261 375.0f, -310.0f, 0.0f, 1.0f,
262 475.0f, -310.0f, 0.0f, 1.0f,
263
264 435.0f, -310.0f, 0.0f, 1.0f,
265 415.0f, -310.0f, 0.0f, 1.0f,
266 415.0f, -200.0f, 0.0f, 1.0f,
267 435.0f, -200.0f, 0.0f, 1.0f,
268
269 //Blue win
270
271 //B
272 -300.0f, 700.0f, 0.0f, 1.0f,
273 -300.0f, 600.0f, 0.0f, 1.0f,
274 -280.0f, 600.f, 0.0f, 1.0f,
275 -280.0f, 700.0f, 0.0f, 1.0f,

```

```

276
277     -250.0f, 700.0f, 0.0f, 1.0f,
278     -250.0f, 600.0f, 0.0f, 1.0f,
279     -230.0f, 600.f, 0.0f, 1.0f,
280     -230.0f, 700.0f, 0.0f, 1.0f,
281
282     -300.0f, 700.0f, 0.0f, 1.0f,
283     -300.0f, 680.0f, 0.0f, 1.0f,
284     -230.0f, 680.f, 0.0f, 1.0f,
285     -230.0f, 700.0f, 0.0f, 1.0f,
286
287     -300.0f, 660.0f, 0.0f, 1.0f,
288     -300.0f, 640.0f, 0.0f, 1.0f,
289     -230.0f, 640.f, 0.0f, 1.0f,
290     -230.0f, 660.0f, 0.0f, 1.0f,
291
292     -300.0f, 620.0f, 0.0f, 1.0f,
293     -300.0f, 600.0f, 0.0f, 1.0f,
294     -230.0f, 600.f, 0.0f, 1.0f,
295     -230.0f, 620.0f, 0.0f, 1.0f,
296
297     //L
298     -200.0f, 700.0f, 0.0f, 1.0f,
299     -200.0f, 600.0f, 0.0f, 1.0f,
300     -180.0f, 600.f, 0.0f, 1.0f,
301     -180.0f, 700.0f, 0.0f, 1.0f,
302
303     -200.0f, 620.0f, 0.0f, 1.0f,
304     -200.0f, 600.0f, 0.0f, 1.0f,
305     -150.0f, 600.f, 0.0f, 1.0f,
306     -150.0f, 620.0f, 0.0f, 1.0f,
307
308     //U
309     -120.0f, 700.0f, 0.0f, 1.0f,
310     -120.0f, 600.0f, 0.0f, 1.0f,
311     -100.0f, 600.f, 0.0f, 1.0f,
312     -100.0f, 700.0f, 0.0f, 1.0f,
313
314     -70.0f, 700.0f, 0.0f, 1.0f,
315     -70.0f, 600.0f, 0.0f, 1.0f,
316     -50.0f, 600.f, 0.0f, 1.0f,

```

```

317 -50.0f, 700.0f, 0.0f, 1.0f,
318
319 -120.0f, 620.0f, 0.0f, 1.0f,
320 -50.0f, 620.0f, 0.0f, 1.0f,
321 -50.0f, 600.0f, 0.0f, 1.0f,
322 -120.0f, 600.0f, 0.0f, 1.0f,
323
324 //E
325 -20.0f, 700.0f, 0.0f, 1.0f,
326 -20.0f, 600.0f, 0.0f, 1.0f,
327 0.0f, 600.f, 0.0f, 1.0f,
328 0.0f, 700.0f, 0.0f, 1.0f,
329
330 -20.0f, 620.0f, 0.0f, 1.0f,
331 30.0f, 620.0f, 0.0f, 1.0f,
332 30.0f, 600.0f, 0.0f, 1.0f,
333 -20.0f, 600.0f, 0.0f, 1.0f,
334
335 -20.0f, 660.0f, 0.0f, 1.0f,
336 30.0f, 660.0f, 0.0f, 1.0f,
337 30.0f, 640.0f, 0.0f, 1.0f,
338 -20.0f, 640.0f, 0.0f, 1.0f,
339
340 -20.0f, 700.0f, 0.0f, 1.0f,
341 30.0f, 700.0f, 0.0f, 1.0f,
342 30.0f, 680.0f, 0.0f, 1.0f,
343 -20.0f, 680.0f, 0.0f, 1.0f,
344
345 //W
346 100.0f, 700.0f, 0.0f, 1.0f,
347 120.0f, 700.0f, 0.0f, 1.0f,
348 140.0f, 600.0f, 0.0f, 1.0f,
349 120.0f, 600.0f, 0.0f, 1.0f,
350
351 140.0f, 600.0f, 0.0f, 1.0f,
352 120.0f, 600.0f, 0.0f, 1.0f,
353 140.0f, 670.0f, 0.0f, 1.0f,
354 160.0f, 670.0f, 0.0f, 1.0f,
355
356 140.0f, 670.0f, 0.0f, 1.0f,
357 160.0f, 670.0f, 0.0f, 1.0f,

```

```

358     180.0f, 600.0f, 0.0f, 1.0f,
359     160.0f, 600.0f, 0.0f, 1.0f,
360
361     180.0f, 600.0f, 0.0f, 1.0f,
362     160.0f, 600.0f, 0.0f, 1.0f,
363     180.0f, 700.0f, 0.0f, 1.0f,
364     200.0f, 700.0f, 0.0f, 1.0f,
365
366     //I
367
368     220.0f, 700.0f, 0.0f, 1.0f,
369     240.0f, 700.0f, 0.0f, 1.0f,
370     240.0f, 600.0f, 0.0f, 1.0f,
371     220.0f, 600.0f, 0.0f, 1.0f,
372
373     //N
374
375     260.0f, 700.0f, 0.0f, 1.0f,
376     280.0f, 700.0f, 0.0f, 1.0f,
377     280.0f, 600.0f, 0.0f, 1.0f,
378     260.0f, 600.0f, 0.0f, 1.0f,
379
380     310.0f, 700.0f, 0.0f, 1.0f,
381     330.0f, 700.0f, 0.0f, 1.0f,
382     330.0f, 600.0f, 0.0f, 1.0f,
383     310.0f, 600.0f, 0.0f, 1.0f,
384
385     280.0f, 700.0f, 0.0f, 1.0f,
386     280.0f, 670.0f, 0.0f, 1.0f,
387     310.0f, 600.0f, 0.0f, 1.0f,
388     310.0f, 630.0f, 0.0f, 1.0f,
389 };
390
391 // culorile varfurilor din culturi
392 GLfloat Colors[] = {
393     0.0f, 1.0f, 0.0f, 1.0f,
394     0.0f, 1.0f, 0.0f, 1.0f,
395     0.0f, 1.0f, 0.0f, 1.0f,
396     0.0f, 1.0f, 0.0f, 1.0f,
397
398     0.5f, 0.5f, 0.5f, 1.0f,

```

```

399 0.5f, 0.5f, 0.5f, 1.0f,
400 0.5f, 0.5f, 0.5f, 1.0f,
401 0.5f, 0.5f, 0.5f, 1.0f,
402
403 1.0f, 1.0f, 1.0f, 1.0f,
404 1.0f, 1.0f, 1.0f, 1.0f,
405 1.0f, 1.0f, 1.0f, 1.0f,
406 1.0f, 1.0f, 1.0f, 1.0f,
407
408     1.0f, 1.0f, 1.0f, 1.0f,
409 1.0f, 1.0f, 1.0f, 1.0f,
410 1.0f, 1.0f, 1.0f, 1.0f,
411 1.0f, 1.0f, 1.0f, 1.0f,
412
413     1.0f, 1.0f, 1.0f, 1.0f,
414 1.0f, 1.0f, 1.0f, 1.0f,
415 1.0f, 1.0f, 1.0f, 1.0f,
416 1.0f, 1.0f, 1.0f, 1.0f,
417
418     1.0f, 1.0f, 1.0f, 1.0f,
419 1.0f, 1.0f, 1.0f, 1.0f,
420 1.0f, 1.0f, 1.0f, 1.0f,
421 1.0f, 1.0f, 1.0f, 1.0f,
422
423     1.0f, 1.0f, 1.0f, 1.0f,
424 1.0f, 1.0f, 1.0f, 1.0f,
425 1.0f, 1.0f, 1.0f, 1.0f,
426 1.0f, 1.0f, 1.0f, 1.0f,
427
428     1.0f, 1.0f, 1.0f, 1.0f,
429 1.0f, 1.0f, 1.0f, 1.0f,
430 1.0f, 1.0f, 1.0f, 1.0f,
431 1.0f, 1.0f, 1.0f, 1.0f,
432
433     1.0f, 1.0f, 1.0f, 1.0f,
434 1.0f, 1.0f, 1.0f, 1.0f,
435 1.0f, 1.0f, 1.0f, 1.0f,
436 1.0f, 1.0f, 1.0f, 1.0f,
437
438     1.0f, 1.0f, 1.0f, 1.0f,
439 1.0f, 1.0f, 1.0f, 1.0f,

```


440	1.0f, 1.0f, 1.0f, 1.0f,
441	1.0f, 1.0f, 1.0f, 1.0f,
442	
443	1.0f, 0.0f, 0.0f, 1.0f,
444	1.0f, 0.0f, 0.0f, 1.0f,
445	1.0f, 0.0f, 0.0f, 1.0f,
446	1.0f, 0.0f, 0.0f, 1.0f,
447	
448	0.0f, 0.0f, 1.0f, 1.0f,
449	0.0f, 0.0f, 1.0f, 1.0f,
450	0.0f, 0.0f, 1.0f, 1.0f,
451	0.0f, 0.0f, 1.0f, 1.0f,
452	
453	1.0f, 1.0f, 0.0f, 1.0f,
454	1.0f, 1.0f, 0.0f, 1.0f,
455	1.0f, 1.0f, 0.0f, 1.0f,
456	1.0f, 1.0f, 0.0f, 1.0f,
457	
458	1.0f, 1.0f, 0.0f, 1.0f,
459	1.0f, 1.0f, 0.0f, 1.0f,
460	1.0f, 1.0f, 0.0f, 1.0f,
461	1.0f, 1.0f, 0.0f, 1.0f,
462	
463	1.0f, 1.0f, 0.0f, 1.0f,
464	1.0f, 1.0f, 0.0f, 1.0f,
465	1.0f, 1.0f, 0.0f, 1.0f,
466	1.0f, 1.0f, 0.0f, 1.0f,
467	
468	1.0f, 1.0f, 0.0f, 1.0f,
469	1.0f, 1.0f, 0.0f, 1.0f,
470	1.0f, 1.0f, 0.0f, 1.0f,
471	1.0f, 1.0f, 0.0f, 1.0f,
472	
473	1.0f, 1.0f, 0.0f, 1.0f,
474	1.0f, 1.0f, 0.0f, 1.0f,
475	1.0f, 1.0f, 0.0f, 1.0f,
476	1.0f, 1.0f, 0.0f, 1.0f,
477	
478	1.0f, 1.0f, 0.0f, 1.0f,
479	1.0f, 1.0f, 0.0f, 1.0f,
480	1.0f, 1.0f, 0.0f, 1.0f,

481	1.0f, 1.0f, 0.0f, 1.0f,
482	
483	1.0f, 1.0f, 0.0f, 1.0f,
484	1.0f, 1.0f, 0.0f, 1.0f,
485	1.0f, 1.0f, 0.0f, 1.0f,
486	1.0f, 1.0f, 0.0f, 1.0f,
487	
488	1.0f, 1.0f, 0.0f, 1.0f,
489	1.0f, 1.0f, 0.0f, 1.0f,
490	1.0f, 1.0f, 0.0f, 1.0f,
491	1.0f, 1.0f, 0.0f, 1.0f,
492	
493	1.0f, 1.0f, 0.0f, 1.0f,
494	1.0f, 1.0f, 0.0f, 1.0f,
495	1.0f, 1.0f, 0.0f, 1.0f,
496	1.0f, 1.0f, 0.0f, 1.0f,
497	
498	1.0f, 1.0f, 0.0f, 1.0f,
499	1.0f, 1.0f, 0.0f, 1.0f,
500	1.0f, 1.0f, 0.0f, 1.0f,
501	1.0f, 1.0f, 0.0f, 1.0f,
502	
503	1.0f, 1.0f, 0.0f, 1.0f,
504	1.0f, 1.0f, 0.0f, 1.0f,
505	1.0f, 1.0f, 0.0f, 1.0f,
506	1.0f, 1.0f, 0.0f, 1.0f,
507	
508	1.0f, 1.0f, 0.0f, 1.0f,
509	1.0f, 1.0f, 0.0f, 1.0f,
510	1.0f, 1.0f, 0.0f, 1.0f,
511	1.0f, 1.0f, 0.0f, 1.0f,
512	
513	1.0f, 1.0f, 0.0f, 1.0f,
514	1.0f, 1.0f, 0.0f, 1.0f,
515	1.0f, 1.0f, 0.0f, 1.0f,
516	1.0f, 1.0f, 0.0f, 1.0f,
517	
518	1.0f, 1.0f, 0.0f, 1.0f,
519	1.0f, 1.0f, 0.0f, 1.0f,
520	1.0f, 1.0f, 0.0f, 1.0f,
521	1.0f, 1.0f, 0.0f, 1.0f,

```

522
523     1.0f, 1.0f, 0.0f, 1.0f,
524     1.0f, 1.0f, 0.0f, 1.0f,
525     1.0f, 1.0f, 0.0f, 1.0f,
526     1.0f, 1.0f, 0.0f, 1.0f,
527
528     1.0f, 1.0f, 0.0f, 1.0f,
529     1.0f, 1.0f, 0.0f, 1.0f,
530     1.0f, 1.0f, 0.0f, 1.0f,
531     1.0f, 1.0f, 0.0f, 1.0f,
532 };
533
534 // se creeaza un buffer nou
535 glGenBuffers(1, &VboId);
536 // este setat ca buffer curent
537 glBindBuffer(GL_ARRAY_BUFFER, VboId);
538 // punctele sunt "copiate" in bufferul curent
539 glBufferData(GL_ARRAY_BUFFER, sizeof(Vertices), Vertices,
540             GL_STATIC_DRAW);
541
542 // se creeaza / se leaga un VAO (Vertex Array Object) - util cand
543 // se utilizeaza mai multe VBO
544 glGenVertexArrays(1, &VaoId);
545 glBindVertexArray(VaoId);
546 // se activeaza lucrul cu attribute; atributul 0 = pozitie
547 glEnableVertexAttribArray(0);
548 glVertexAttribPointer(0, 4, GL_FLOAT, GL_FALSE, 0, 0);
549
550 // un nou buffer, pentru culoare
551 glGenBuffers(1, &ColorBufferId);
552 glBindBuffer(GL_ARRAY_BUFFER, ColorBufferId);
553 glBufferData(GL_ARRAY_BUFFER, sizeof(Colors), Colors,
554             GL_STATIC_DRAW);
555 // atributul 1 = culoare
556 glEnableVertexAttribArray(1);
557 glVertexAttribPointer(1, 4, GL_FLOAT, GL_FALSE, 0, 0);
558 }
559 void DestroyVBO(void)
560 {
561     glDisableVertexAttribArray(1);
562     glDisableVertexAttribArray(0);

```

```

560     glBindBuffer(GL_ARRAY_BUFFER, 0);
561     glDeleteBuffers(1, &ColorBufferId);
562     glDeleteBuffers(1, &VboId);
563     glBindVertexArray(0);
564     glDeleteVertexArrays(1, &VaoId);
565 }
566
567 void CreateShaders(void)
568 {
569     ProgramId = LoadShaders("03_02_Shader.vert", "03_02_Shader.frag");
570     glUseProgram(ProgramId);
571 }
572 void DestroyShaders(void)
573 {
574     glDeleteProgram(ProgramId);
575 }
576
577 void Initialize(void)
578 {
579     glClearColor(1.0f, 1.0f, 1.0f, 0.0f); // culoarea de fond a
        ecranului
580     CreateVBO();
581     CreateShaders();
582     codColLocation = glGetUniformLocation(ProgramId, "codCuloare");
583     myMatrixLocation = glGetUniformLocation(ProgramId, "myMatrix");
584 }
585 void RenderFunction(void)
586 {
587     glClear(GL_COLOR_BUFFER_BIT);
588
589     // TO DO: schimbati transformarile (de exemplu deplasarea are loc
        pe axa Oy sau pe o alta dreapta)
590     resizeMatrix = glm::ortho(-width, width, -height, height); //
        scalam, "aducem" scena la "patratul standard" [-1,1]x[-1,1]
591     matrTransl = glm::translate(glm::mat4(1.0f), glm::vec3(i, k, 0.0))
        ; // controleaza translatia de-a lungul lui Ox
592     matrTransl3 = glm::translate(glm::mat4(1.0f), glm::vec3(0.0, h,
        0.0));
593     matrDepl = glm::translate(glm::mat4(1.0f), glm::vec3(1.0, 1.0,
        0.0)); // plaseaza patratul rosu

```

```

594     matrScale2 = glm::scale(glm::mat4(1.0f), glm::vec3(1.0, 1.0, 0.0))
        ; // folosita la desenarea patraturului rosu
595     matrTransl2 = glm::translate(glm::mat4(1.0f), glm::vec3(j, 1, 0.0)
        );
596     matrRot = glm::rotate(glm::mat4(1.0f), angle, glm::vec3(0.0, 0.0,
        1.0)); // rotatie folosita la deplasarea patraturului rosu
597
598     // Matricea de redimensionare (pentru elementele "fixe")
599     myMatrix = resizeMatrix;
600     // Culoarea
601     codCol = 0;
602     // Transmitere variabile uniforme
603     glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0])
        ;
604     glUniform1i(codColLocation, codCol);
605
606     glDrawArrays(GL_POLYGON, 0, 4);
607     glDrawArrays(GL_POLYGON, 4, 4);
608     glDrawArrays(GL_POLYGON, 8, 4);
609     glDrawArrays(GL_POLYGON, 12, 4);
610     glDrawArrays(GL_POLYGON, 16, 4);
611     glDrawArrays(GL_POLYGON, 20, 4);
612     glDrawArrays(GL_POLYGON, 24, 4);
613     glDrawArrays(GL_POLYGON, 28, 4);
614     glDrawArrays(GL_POLYGON, 32, 4);
615     glDrawArrays(GL_POLYGON, 36, 4);
616     glDrawArrays(GL_POLYGON, 48, 4);
617     glDrawArrays(GL_POLYGON, 52, 4);
618     glDrawArrays(GL_POLYGON, 56, 4);
619     glDrawArrays(GL_POLYGON, 60, 4);
620     glDrawArrays(GL_POLYGON, 64, 4);
621     glDrawArrays(GL_POLYGON, 68, 4);
622     glDrawArrays(GL_POLYGON, 72, 4);
623     glDrawArrays(GL_POLYGON, 76, 4);
624     glDrawArrays(GL_POLYGON, 80, 4);
625     glDrawArrays(GL_POLYGON, 84, 4);
626     glDrawArrays(GL_POLYGON, 88, 4);
627     glDrawArrays(GL_POLYGON, 92, 4);
628     glDrawArrays(GL_POLYGON, 96, 4);
629     glDrawArrays(GL_POLYGON, 100, 4);
630     glDrawArrays(GL_POLYGON, 104, 4);

```

```

631     glDrawArrays(GL_POLYGON, 108, 4);
632
633
634
635     // Matricea pentru dreptunghiul rosu
636     myMatrix = resizeMatrix * matrTransl * matrDepl * matrScale2;
637     // Culoarea
638     codCol = 2;
639     // Transmite variabile uniforme
640     glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0])
        ;
641     glUniform1i(codColLocation, codCol);
642     // Apelare DrawArrays
643     glDrawArrays(GL_POLYGON, 40, 4);
644
645     // Matricea pentru dreptunghiul rosu
646     myMatrix = resizeMatrix * matrTransl2 * matrDepl * matrScale2 *
        matrRot;
647     // Culoarea
648     codCol = 1;
649     // Transmite variabile uniforme
650     glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0])
        ;
651     glUniform1i(codColLocation, codCol);
652     // Apelare DrawArrays
653     glDrawArrays(GL_POLYGON, 44, 4);
654
655
656     myMatrix = resizeMatrix * matrTransl3;
657     glUniformMatrix4fv(myMatrixLocation, 1, GL_FALSE, &myMatrix[0][0])
        ;
658     glUniform1i(codColLocation, codCol);
659     glDrawArrays(GL_POLYGON, 112, 4);
660     glDrawArrays(GL_POLYGON, 116, 4);
661     glDrawArrays(GL_POLYGON, 120, 4);
662     glDrawArrays(GL_POLYGON, 124, 4);
663     glDrawArrays(GL_POLYGON, 128, 4);
664     glDrawArrays(GL_POLYGON, 132, 4);
665     glDrawArrays(GL_POLYGON, 136, 4);
666     glDrawArrays(GL_POLYGON, 140, 4);
667     glDrawArrays(GL_POLYGON, 144, 4);

```

```

668     glDrawArrays(GL_POLYGON, 148, 4);
669     glDrawArrays(GL_POLYGON, 152, 4);
670     glDrawArrays(GL_POLYGON, 156, 4);
671     glDrawArrays(GL_POLYGON, 160, 4);
672     glDrawArrays(GL_POLYGON, 164, 4);
673     glDrawArrays(GL_POLYGON, 168, 4);
674     glDrawArrays(GL_POLYGON, 172, 4);
675     glDrawArrays(GL_POLYGON, 176, 4);
676     glDrawArrays(GL_POLYGON, 180, 4);
677     glDrawArrays(GL_POLYGON, 184, 4);
678     glDrawArrays(GL_POLYGON, 188, 4);
679     glDrawArrays(GL_POLYGON, 192, 4);
680     glDrawArrays(GL_POLYGON, 196, 4);
681     glDrawArrays(GL_POLYGON, 200, 4);
682     glutSwapBuffers();
683     glFlush();
684 }
685 void Cleanup(void)
686 {
687     DestroyShaders();
688     DestroyVBO();
689 }
690
691 int main(int argc, char* argv[])
692 {
693     glutInit(&argc, argv);
694     glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
695     glutInitWindowPosition(100, 100);
696     glutInitWindowSize(1000, 700);
697     glutCreateWindow("Proiect 1 - Depasire intre 2 dreptunghiuri");
698     glewInit();
699     Initialize();
700     glutDisplayFunc(RenderFunction);
701     glutMouseFunc(mouse);
702     glutCloseFunc(Cleanup);
703     glutMainLoop();
704 }

```

Referințe

- Fișierul *03_02_animatie_new.cpp* din *Laborator 3*.
- *Materialele din Curs*.