აკუმულაციის ბუფერი

წინა ლექციაზე ჩვენ ვისწავლეთ სტენსილ ბუფერის გამოყენება სხვადასხვა ეფექტების მისაღებად. ამ ლექციაზე შევიწავლოთ აკუმულაციის ბუფერს. აკუმულაციის ბუფერი გამოყენება ისეთი ეფექტების მისაღებად როგორიცაა: მთლიანი სცენის ალიასინგი (დაკბილული კუთხური-ზედაპირების მოცილება), გადღაბნილი მოძრაობა (Motion Blur), რბილი ჩრდილები, სიღმისეული ეფექტი (Depth of Field) და სხვა.

გრაფიკული ოპერაციების შედეგი პირდაპირ არ აისახება აკუმულაციის ბუფერში. როგორც წესი საბოლოო გამოსახულება ფორმირდება ფერის ბუფერში, ამის შემდეგ ხდება რამოდენიმე გამოსახულების "დაგროვება" აკუმულაციის ბუფერში სხვადასხვა ეფექტის მისაღებად. აღსანიშნავია რომ ოპერაციებიის სერიის დასრულების შემდეგ აკუმულაციის ბუფერში ჩაწერილ ნახატს შესაძლებელია აღმოაჩდეს უფრო მეტი ფერის გარჩევადობა (მეტი ბიტი თითოეულ ფერის კომპონენტაზე) ვიდრე საწყის ნახატებს. როგორც უკვე მიხვდით, აკუმულაციის ბუფერის გამოყენებისას სხვადასხვა ეფექტის მიღების მიზნით, გვიწევს სცენის რამოდენიმეჯერ დახატვა, შესაბამისად ხდება წარმადობის დაწევა.

მარტივად რომ ვთქვათ აკუმულაციის ბუფერის გამოყენებისას ჩვენ საშუალება გვაქვს, ავიღოთ რამოდენიმე დახატული კადრი და მათი ერთმანეთზე დადებით ან რაიმე სხვა კომბინაციით მივიღოთ ახალი ეფექტი.

განვიხილოთ ფუნქცია : **glAccum(GLenum op, GLfloat value)** ამ ფუნქციის გამოყენებით ხდება აკუმულაციის ბუფერის მართვა. პირველი პარამეტრი შესაძლებელია იყოს: GL_ACCUM, GL_LOAD, GL_RETURN, GL_ADD და GL_MULT. განვიხილოთ თითოეული მათგანი დაწვრილებით.

GL_ACCUM - კითხულობს პიქსელებს მიმდინარე ბუფერიდან და ახდენს თითოეული ფერის კომპონენტის (R,G,B,A) გამრავლებას value პარამეტრზე. მიღებული შედეგები იჯამება აკუმულაციის ბუფერში.

GL_LOAD - მუშაობს როგორც GL_ACCUM ერთი განსხვავებით, მიღებული შედეგები არ იჯამება, ხდება შესაბამისი მნიშვნელობების ჩანაცვლება ახლით.

GL_RETURN - ამ დროს ხდება მნიშვნელობების ამოღება აკუმულაციის ბუფერიდან და მათი გადატანა მიმდინარე კადრის (ან სხვა აქტიურ) ბუფერში.

GL_ADD და GL_MULT - დროს ხდება აკუმულაციის ბუფერის მნიშვნელობების დამატება ან გამრავლება value პარამეტრზე. GL_MULT-ის დროს ხდება მნიშვნელობების დაყვანა [-1.0, 1.0] დიაპაზონზე. GL_ADD-ის დროს ასეთ დაყვანა არ ხდება.

განვიხილოთ აკუმულაციის ბუფერის გამოყენების რამოდენიმე კონკრეტული მაგალითი.

მთლიანი სცენის ალიასინგი

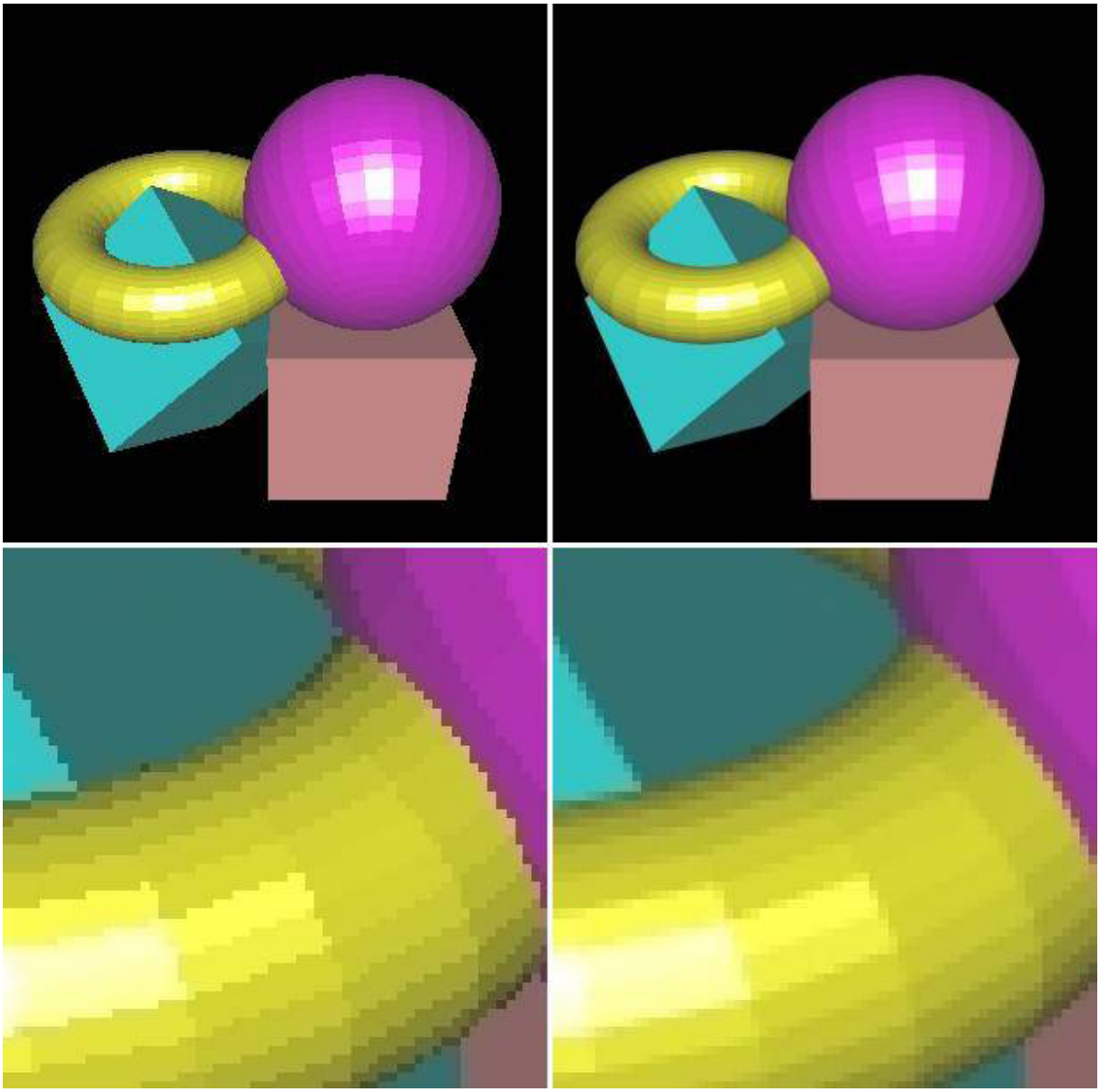
ამ ეფექტის მისაღებად საჭიროა

1. გასუფთავდეს აკუმულაციის ბუფერი. `glClear(GL_ACCUM_BUFFER_BIT);`

2. n-ჯერ დაიხატოს სცენა, მცირედი წანაცვლებით აკუმულაციის ბუფერში glAccum(GL_ACCUM, 100/n).

3. მოხდეს სცენის გადატანა კადრის ბუფერში glAccum(GL_RETURN, 1.0)

ამ ეფექტის მუშაობის შედეგი და საწყისი მონაცემები შეგიძლიათ იხილოთ შემდეგ ნახატზე.



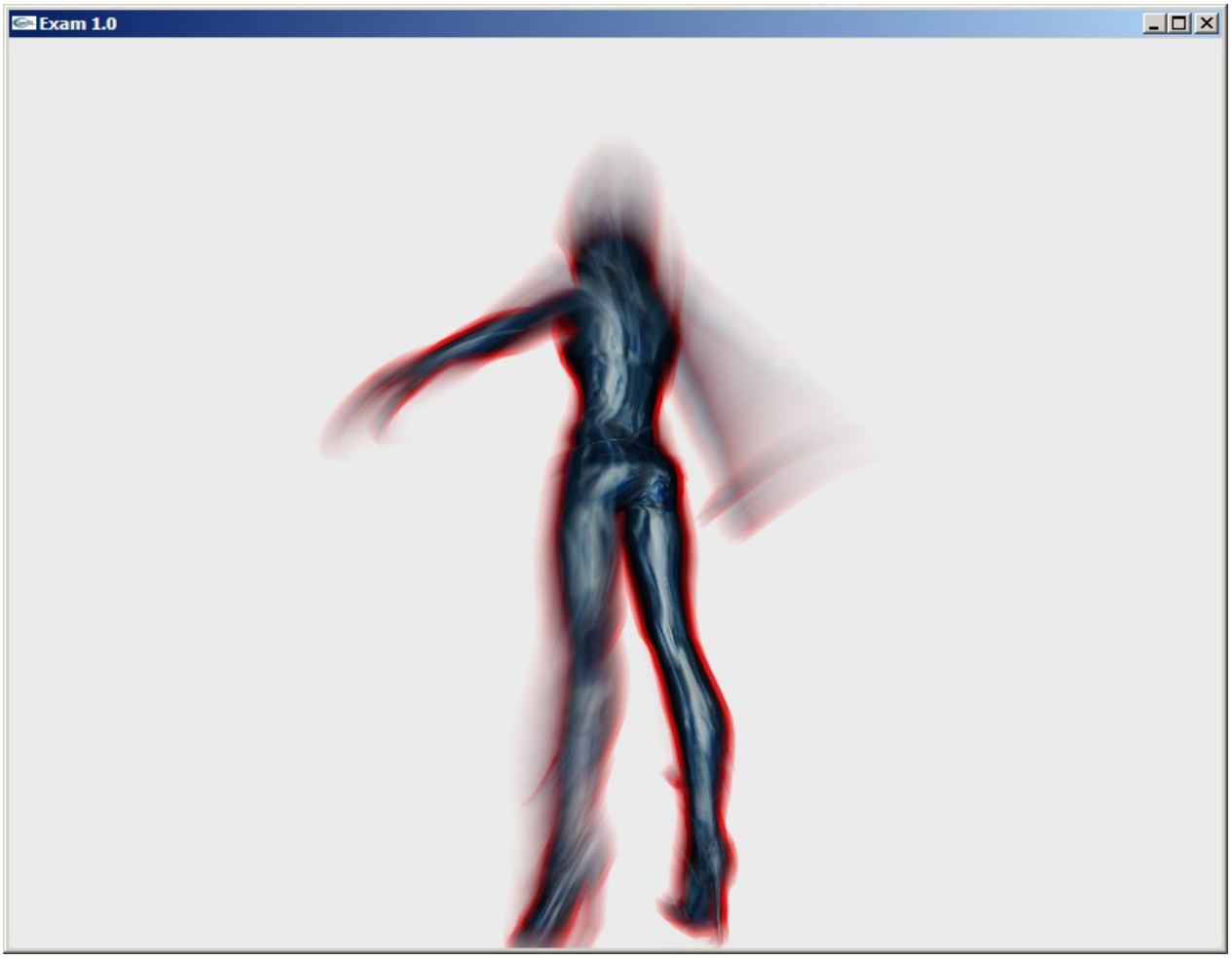
```
glClear(GL_ACCUM_BUFFER_BIT);
for (i=0;i<N;i++)
{
glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
```

//აქ ხდება სცენის მცირე კუთხით წანაცვლება/მობრუნება

```
 displayObjects();
 glAccum(GL_ACCUM,1.0/N);
}

glAccum(GL_RETURN,1.0);
glFlush();
```

გადღაბნილი მოძრაობა (Motion Blur)



ამ ეფექტის მისაღწევად საჭიროა სცენის რენდერირებისას დავამატოთ შემდეგი კოდი:

```
float mblur_factor = 0.98f;

if (mblur_factor > 0){
	bf = powf(mblur_factor, 0.5);
	glAccum(GL_MULT, mblur_factor);
	glAccum(GL_ACCUM, 1.0f - mblur_factor);
	glAccum(GL_RETURN, 1);
}
```

mblur_factor - ის მნიშვნელობა შესაძლებელია იცვლებოდეს 0.9-დან 1.0-მდე. რაც უფრო მაღალია მნიშვნელობა მით უფრო დიდხანს იმოქმედებს გადღაბნილი მოძრაობის ეფექტი სცენაზე.

სცენის ობიექტების არჩევა

გრაფიკული პროგრამის ერთადერთი მიზანი არ არის 3D ობიექტების ეკრანზე დახატვა და ანიმაცია. ასეთი ტიპის პროგრამების უმეტესობა მოითხოვს ინტერაქტიულობას, ხანდახან ამ დახატული ობიექტების არჩევას, მონიშვნას და მანიპულირებას. სწორედ ასეთი ოპერაციების ჩატარების მიზნით OpenGL-ში შემოიღეს ე.წ. არჩევის მექანიზმი(selection mechanism) და მითითების რეჟიმი(picking).

სინამდვილეში არჩევის რეჟიმი ესაა OpenGL-ის ერთერთი მდგომარეობა მეორენაირად უკუკავშირის(feedback) რეჟიმი. ამ დროს ჩვეულებრივად ხდება ვიზუალიზაციისთვის საჭირო გათვლები ერთი განსხვავებით, ეკრანზე არ ხდება დახატვა. გამოთვილი შედეგები ეკრანის მაგივრად უბრუნდება მომხმარებლის პროგრამას. მაგალითად თუ პროგრამას უნდა 3D სცენა დახატოს პლოტერზე, მაშინ შესაძლებელია ეს მოხდეს უკუკავშირის რეჟიმში და მიღებული მონაცემები პროგრამულად გარდაიქმნას პლოტერის ბრძანებებში. დეტალურად განვიხილოთ ეს რეჟიმები.

არჩევის რეჟიმი

როგორც წესი არჩევის რეჟიმში გადასვლამდე იხატება სცენა. ეს ხდება იმიტომ რომ არჩევის რეჟიმის დროს არ ხდება სცენის გადახატვა მანამ სანამ არ გავთიშავთ ამ რეჟიმს. არჩევის რეჟიმის გათიშვის მერე პროგრამას უბრუნდება ყველა იმ პრიმიტივის სია რომელიც მოხვდა ფრუსტუმში. თითოეული პრიმიტივი რომელიც ხვდება ხედვის არეში იწვევს ე.წ. ჰიტს(hits). სინამდვილეში პროგრამა იღებს სიას რომელიც შედგება წყვილებისგან, ობიექტის სახელი - და ჰიტების სია (hiteroids), რომელიც შეესაბამება მიმიდინარე სახელების სტეკს. სახელების სტეკი ივსება ხატვის დროს. ანუ ხდება სახელის დამატება სახელების სტეკში, პრიმიტივის დახატვა, შემდეგი სახელი შემდეგი პრიმიტივი და ა.შ. ეს საშუალებას აძლებს OpenGL-ს გააკეთოს სცენის ობიექტების იდენტიფიცირება არჩევის რეჟიმის დროს. ამასთან ერთად არსებობს ფუნქციები რომელიც აადვილებს კურსორთან ახლოს მყოფი ობიექტების არჩევას(picking).

არჩევის რეჟიმის ჩასართველად საჭიროა შემდეგი ოპერაციების ჩატარება:

1. **glSelectBuffer()**-ის საშუალებით ხდება უკან დაბრუნებული ინფორმაციის შემნახველია მასივის დანიშვნა.
2. არჩევის რეჟიმში გადასვლა **glRenderMode(GL_SELECT)**
3. სახელების სტეკის ინიციალიზაცია და სახელის სტეკში ჩადება - **glInitNames, glPushName**
4. სცენის ხედვის მოცულობის დაზუსტება. როგორც წესი ეს განსხავდება ხატვის სცენის მოცულობისგან.
5. სცენის ობიექტების დახატვა და სტეკში სახელების გადაცემა.
6. არჩევის რეჟიმიდან გამოსვლა და ჰიტების დამუშავება.

დეტალურად განვიხილოთ თითოეული ეტაპი:

void glSelectBuffer (GLsizei size, GLuint* buffer);

როგორ აღვნიშნეთ ამ ფუნქციის საშუალებით ხდება ჰიტების მასივის მიწოდება, სადაც აკუმულირდება ინფორმაცია არჩევის რეჯიმიდან გამოსვლის შემდეგ. Size-ის პარამეტრი მიუთითებს მაქსიმალურ ზომაზე.

void glRenderMode (GLenum mode);

mode - პარამეტრი შეიძლება იყოს:
GL_SELECTION (არჩევის რეჟიმში გადასვლა),
GL_RENDER(ხატვის რეჟიმი)
GL_FEEDBACK(უკუკავშირის რეჟიმი). უკუკავშირის რეჟიმში გადასვლამდე ინფორმაციის მასივი მიეწოდება glFeedbackBuffer() - ფუნქციის საშუალებით.
მიმდინარე რეჟიმის გასაგებად გამოიყენეთ GL_RENDER_MODE პარამეტრი ფუნქციაში glGetIntegerv()

სცენის ობიექტების იდენტიფიცირებისთვის გვჭირდება სახელების სტეკი. მისი ინიციალიზაცია ხდება glInitNames() ფუნქციის საშუალებით. ეს ფუნქცია ასუფთავებს სახელების სტეკს. ქვემოთ მოყვანილია სახელების სტეკთან მუშაობის მაგალითი:

```
glInitStack();
glPushName(0);

glPushMatrix();
glLoadName(1);
//ვხატავთ რაიმე ობიექტს1
glLoadName(2);
//ვხატავთ სხვა სცენის ობიექტ2
glLoadName(3);
//ვხატავთ სცენის ობიექტ3
//ვხატავთ სცენის ობიექტ4
glPopMatrix();
```

ზემოთ მოყვანილ მაგალითში 1-2 სცენის ობიექტს საკუთარი უნიკალური სახელბი აქვთ. 3-4 ობიექტებს კი საზიარო. გაითვალისწინეთ რომ სახელების სტეკის ზომა შეზღუდულია. როგორც წესი ეს ზომა არ აღემატება 64-ს. მოყვანილ მაგალითში ხდება სტეკის გასუფთავება glInitNames-საშუალებით. შემდეგ ხდება სტეკში სახელის დამატება glPushName(0). ეს აუცილებელია იმიტომ რომ არ მოხდეს შეცდომა glLoadName-ს გამოძახებისას(როდესაც სტეკი ცარიელია). თუ არჩევის რეჟიმი არ არის გააქტიურებული მოხდება ყველა ზემოთ ჩამოთვლილი ფუნქციების გამოძახებების იგნორირება.

თითოეული ჰიტის ჩანაწერი შედგენა 4 კომპონენტისგან, ესენია (ჩამოთვილია მიმდევრობით):

1. სახელების რაოდენობა სტეკში მიმდინარე ჰიტის დაფიქსირებისას.
2. მაქსიმალური და მინიმალური Z კოორდინატა (ფანჯრის). ეს მნიშვნელობა განზაღვრულია [0,1] დიაპაზონში, მრავლდება 2 (ხარისხად 32) - 1 -ზე და მრგვალდება უახლოეს უნიშნო მთელამდე.
3. სახელების სტეკის მდგომარეობა მიმდინარე ჰიტის დროს.