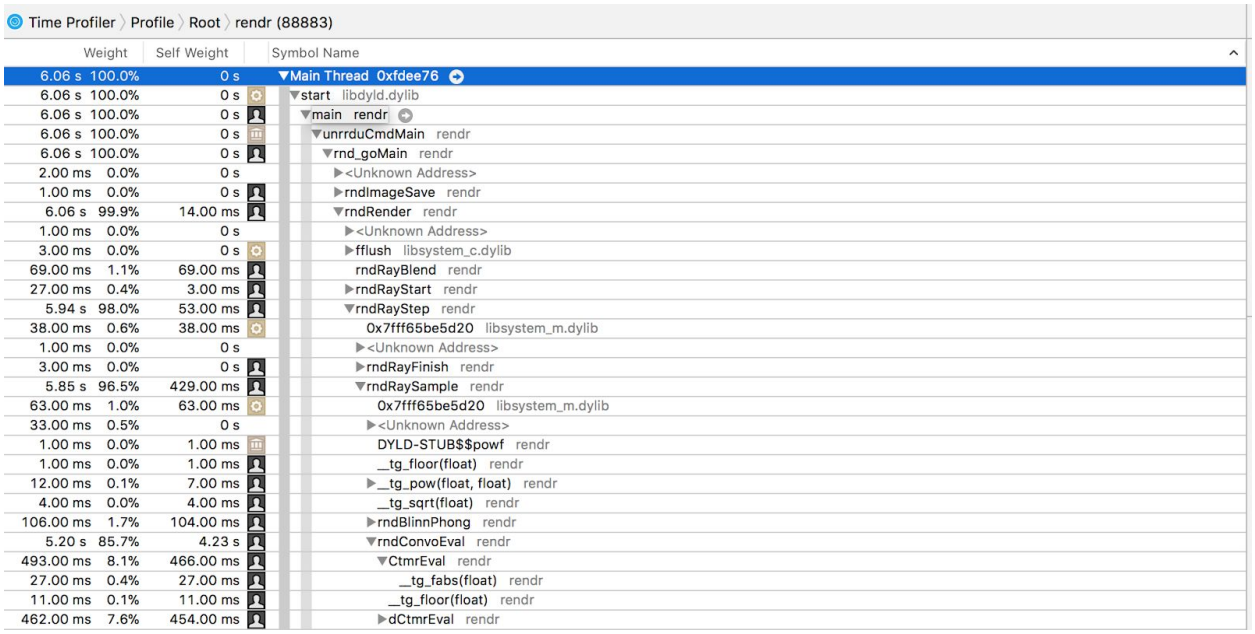
P3 phase 3 Write-up

The example I used to test render go is the one to produce cube-rgb.nrrd. After running the command:

./rendr go -petc $CUBE -fov 14 -us 0.03 -s 0.03 -k ctmr -p rgbalit -b over -lut lut.nrrd -lit $SCIVIS/lit/rgb.txt -o cube-rgb.nrrd

using Time Profiler, I noticed in the call tree as shown below that the bottleneck was from the function **rndConvoEval**, which took 5.20 seconds out of the total rendr go running time 6.06 seconds:



the call tree before code change

Then I checked the **rndConvoEval** function in convo.c, and found possible code fragments for improvement:

1) three for loops for checking the cnv->inside is true or false which should be combined into one:

```
for(int i = ctx->lower;i<=ctx->upper;i++){
  if(i+n3<0||i+n3>(signed)(ctx->vol->size[2])-1
    ||i+n2<0||i+n2>(signed)(ctx->vol->size[1])-1
    ||i+n1<0||i+n1>(signed)(ctx->vol->size[0])-1){
          cnv->inside = 0;
    }
}
```

2) two 3-layer embedded loops dealing with convolution and gradient respectively which should be combined into one 3-layer embedded loops:

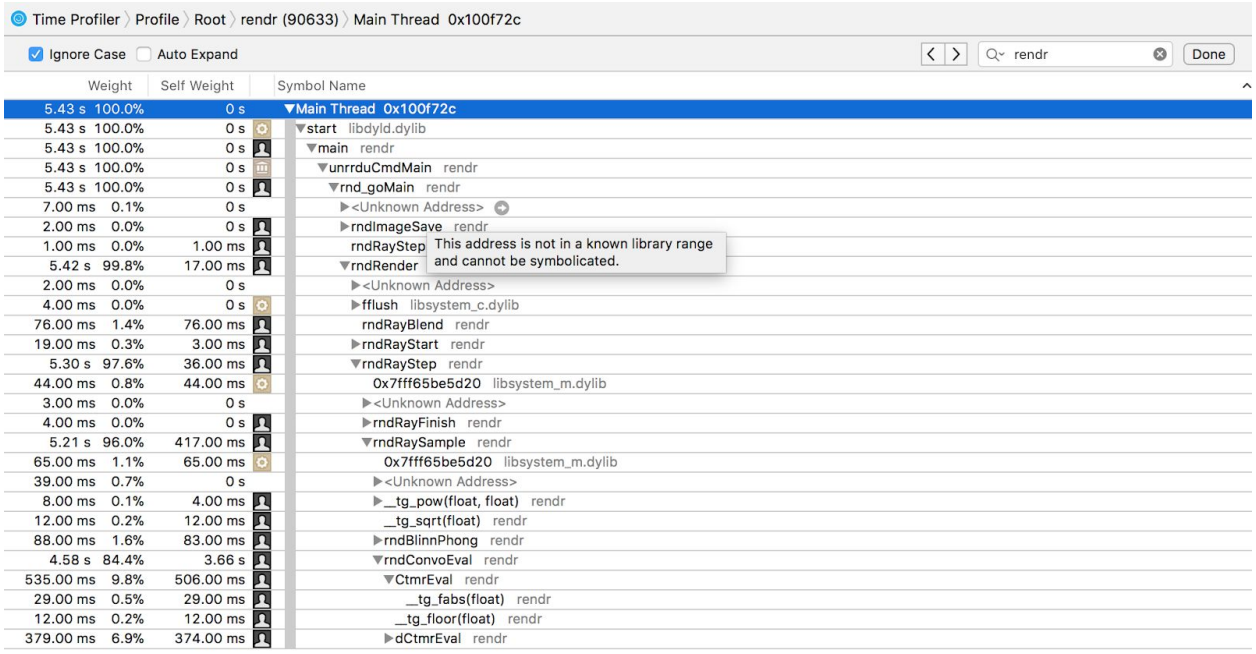
```
for(int i3 = ctx->lower,j3=0;i3<=ctx->upper;i3++,j3++)
 for(int i2 = ctx->lower,j2=0;i2<=ctx->upper;i2++,j2++)
   for(int i1 = ctx->lower,j1=0;i1<=ctx->upper;i1++,j1++){

     int index_1D = (n3+i3)*(ctx->vol->size[1]*ctx->vol->size[0])+
                    (n2+i2)*(ctx->vol->size[0])+n1+i1;
     if(ctx->vol->dtype == rndTypeFloat)
       cov_sum += ctx->vol->data.fl[index_1D]*cnv->kvalues1[j1]*cnv->kvalues2[j2]*cnv->kvalues3
     else if(ctx->vol->dtype == rndTypeShort)
       cov_sum += ctx->vol->data.ss[index_1D]*cnv->kvalues1[j1]*cnv->kvalues2[j2]*cnv->kvalues3
      if (cnv->grad_need){
       if(ctx->vol->dtype == rndTypeFloat){
       cov_sum_deriv1 += ctx->vol->data.fl[index_1D]*cnv->the_kvalues1[j1]*cnv->kvalues2[j2]*cn
       cov_sum_deriv2 += ctx->vol->data.fl[index_1D]*cnv->kvalues1[j1]*cnv->the_kvalues2[j2]*cn
       cov_sum_deriv3 += ctx->vol->data.fl[index_1D]*cnv->kvalues1[j1]*cnv->kvalues2[j2]*cnv->t
       } else if(ctx->vol->dtype == rndTypeShort){
       cov_sum_deriv1 += ctx->vol->data.ss[index_1D]*cnv->the_kvalues1[j1]*cnv->kvalues2[j2]*cn
       cov_sum_deriv2 += ctx->vol->data.ss[index_1D]*cnv->kvalues1[j1]*cnv->the_kvalues2[j2]*cn
       cov_sum_deriv3 += ctx->vol->data.ss[index_1D]*cnv->kvalues1[j1]*cnv->kvalues2[j2]*cnv->t
     }

 }

}
```

then running the same command use Time Profiler, and the function **rndConvoEval** took 4.58 seconds out of the total rendr go running time 5.43 seconds:



the call tree after code change

Comparing two total running time before and after code change, the improvement 6.06-5.43 = **0.63(s)**, which approximately equals to the running time improvement from the function **rndConvoEval** 5.20-4.58 = **0.62(s),** which indicates that the code change on the function **rndConvoEval** effectively eliminate the bottleneck.