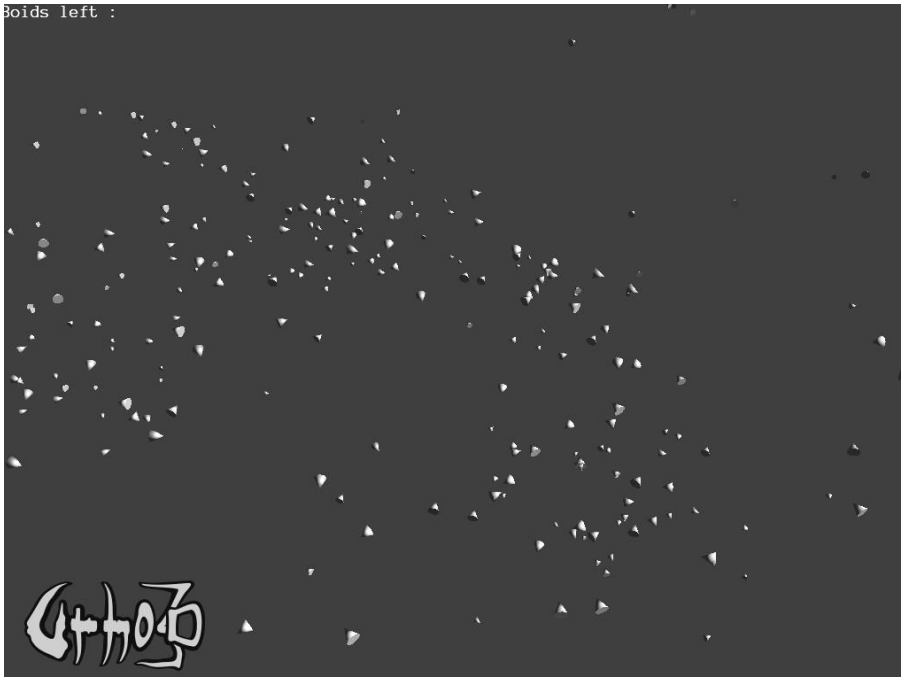
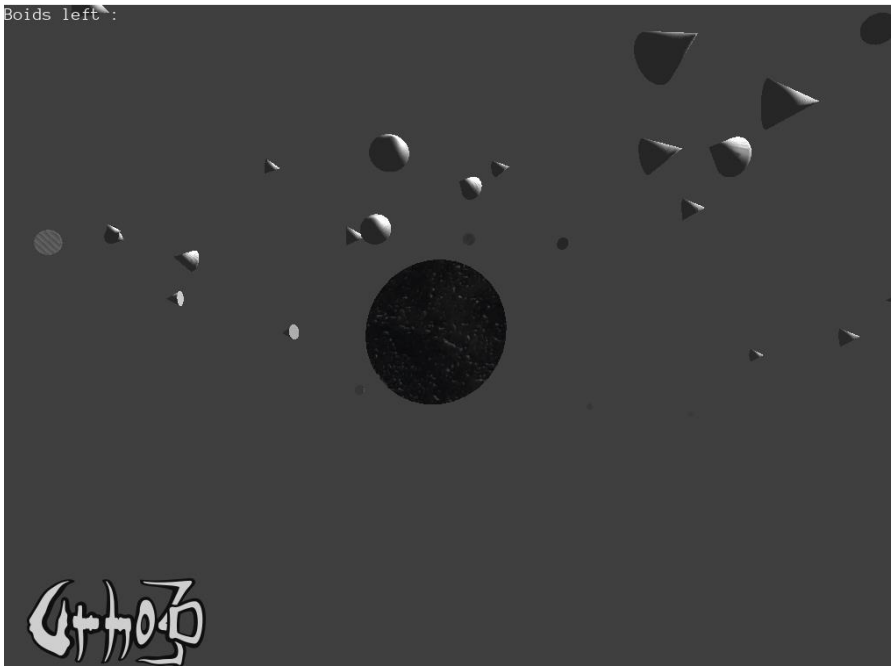
CGP3016M
Advanced Games Programming
Joe Martin – S15596040

Contents

Game Description

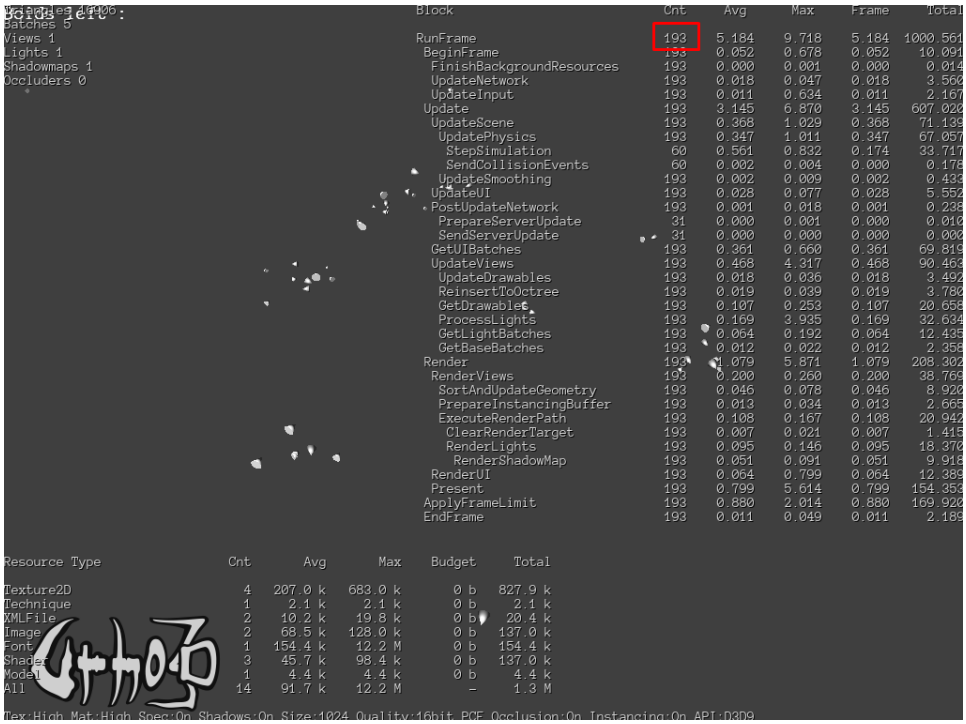
The game I created, is a multiplayer space game using the Urho3D engine. The game allowed multiple players, all with their own ships, which can fire a bullet. The game has 300 other spaceships that fly around in swarms (boids). The objective of the game is to destroy as many ships as you can. You can do this be either flying into then, or by shooting them with your bullet. They will then be removed from the game. The players ship can move forward, turn, and move up and down. The direction of the ship will also turn to face the direction it is moving.



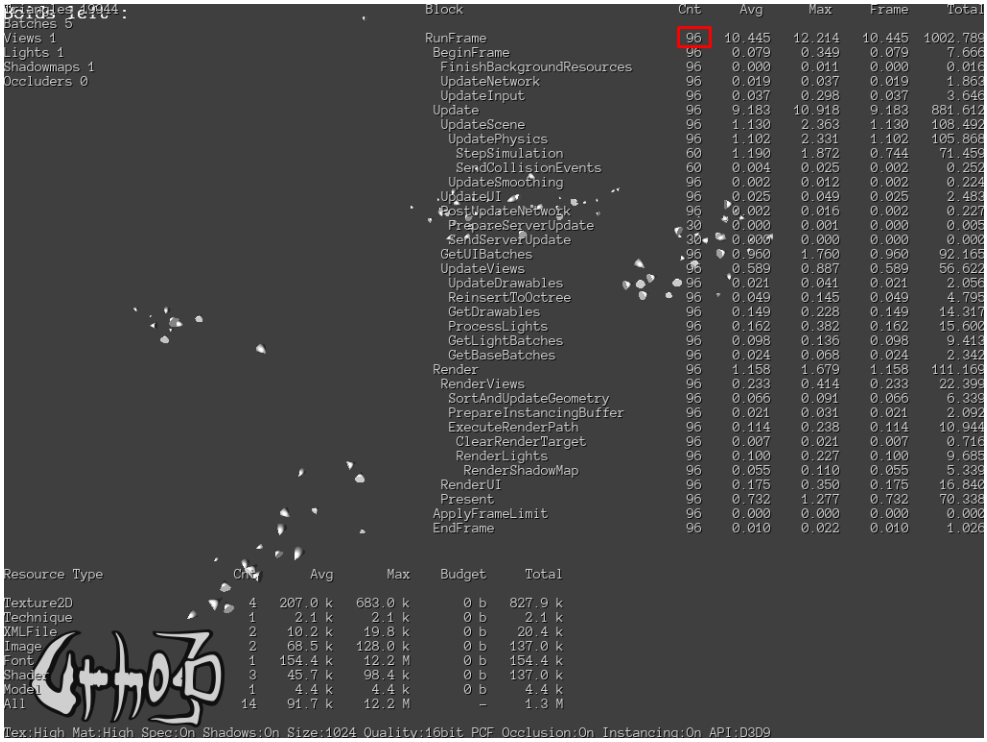


Analysis

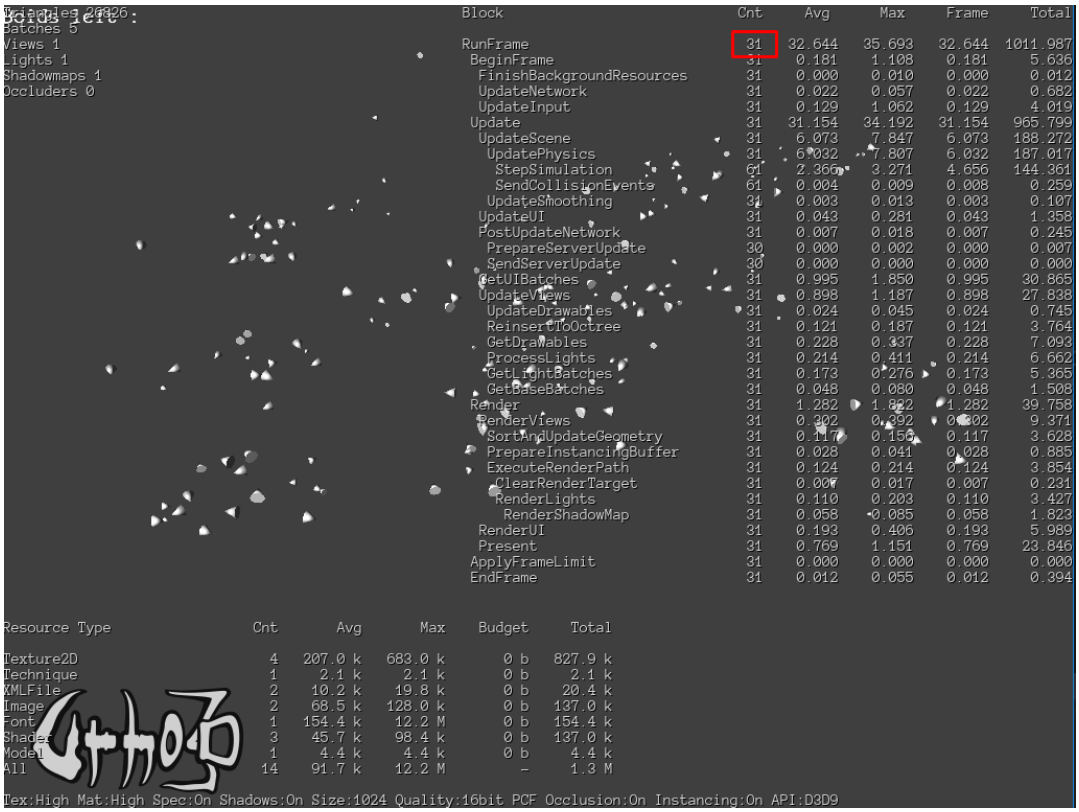
50 boids, 193 frames:



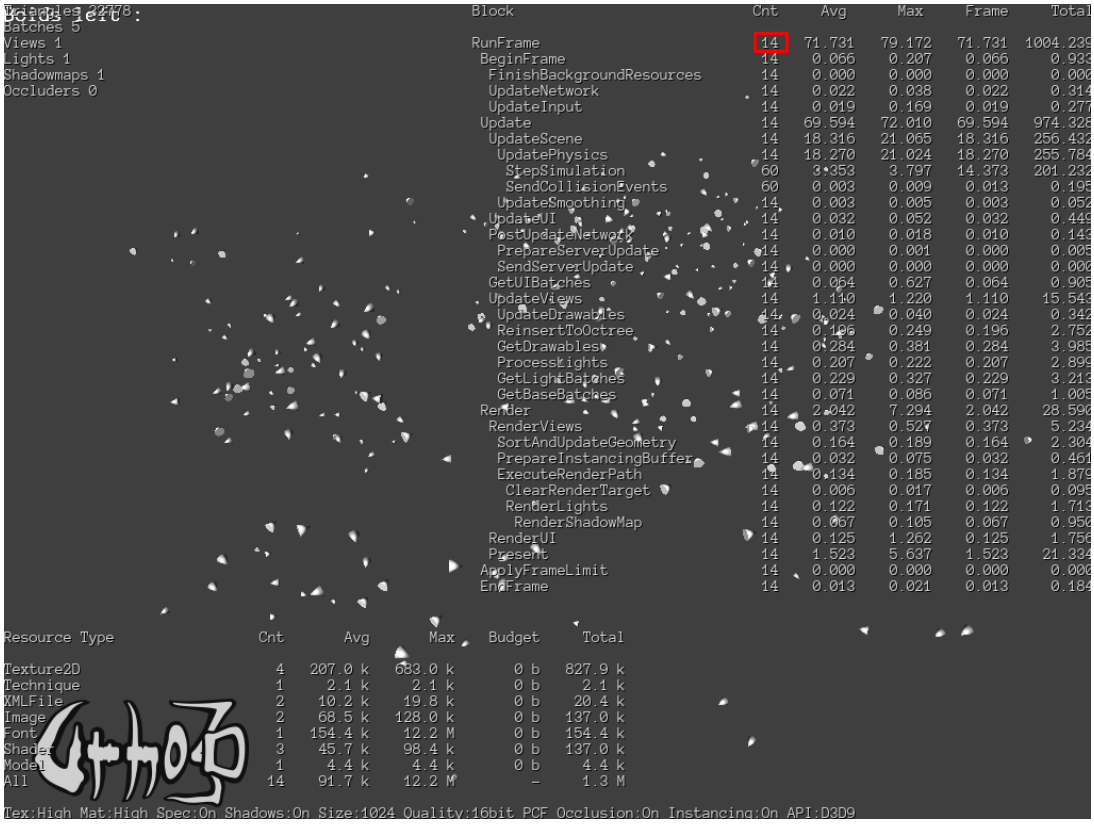
100 boids, 96 frames:



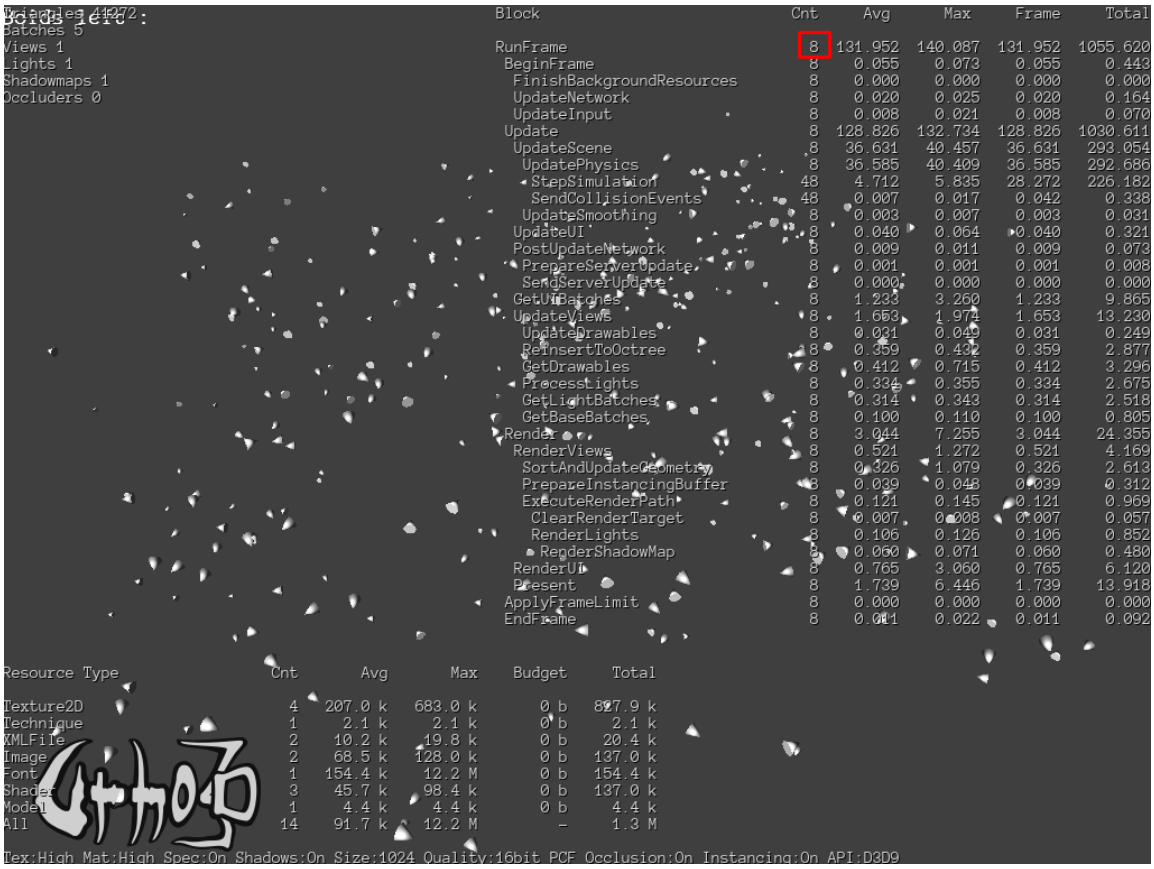
200 boids, 31 frames:



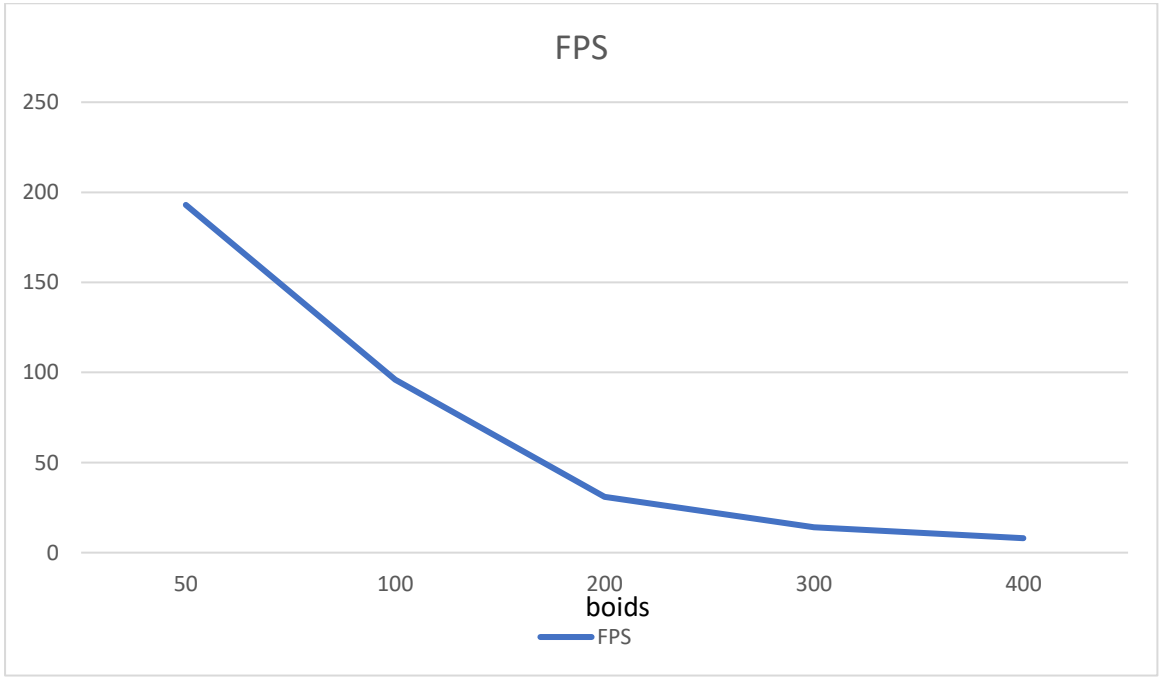
300 boids, 14 frames:



400 boids, 8 frames:



Graph

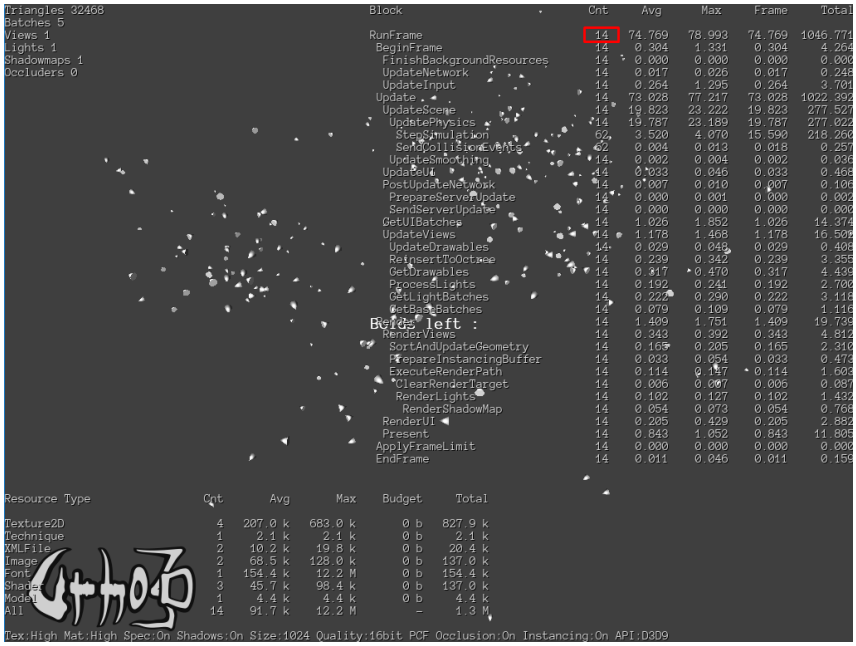


Optimisation

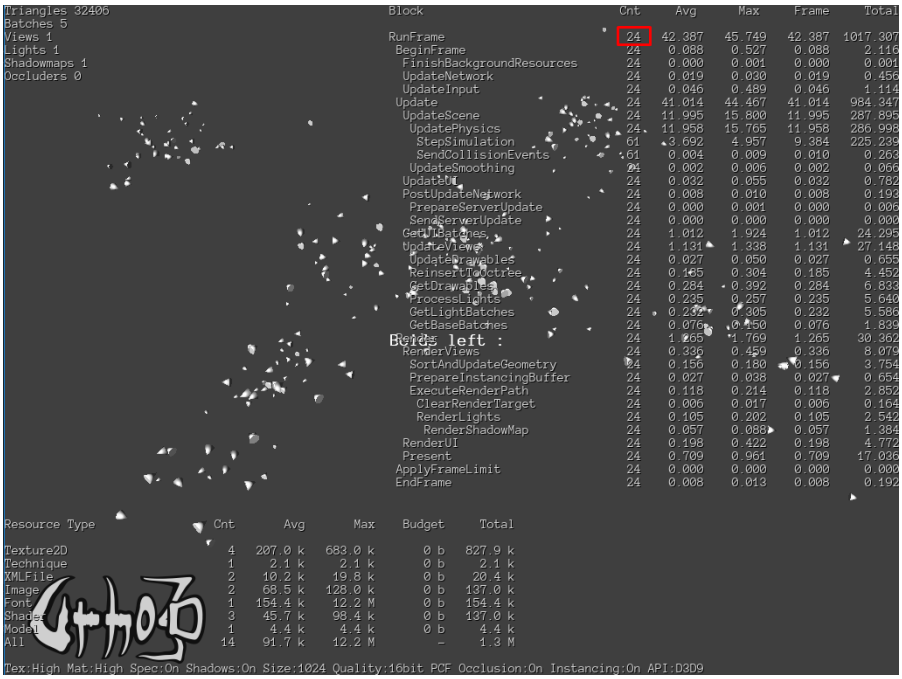
During this optimisation stage, I am going to carry out multiple experiments, using several different methods:

Boid Groups

The first method I will use, is Boid Splitting. This is when I take the boids and split them up into smaller groups. Initially, I have one big group of 300 boids. This means that all 300 boids are interacting with each other, applying forces, and searching for each other. This is very intensive and can cause low frames (14):

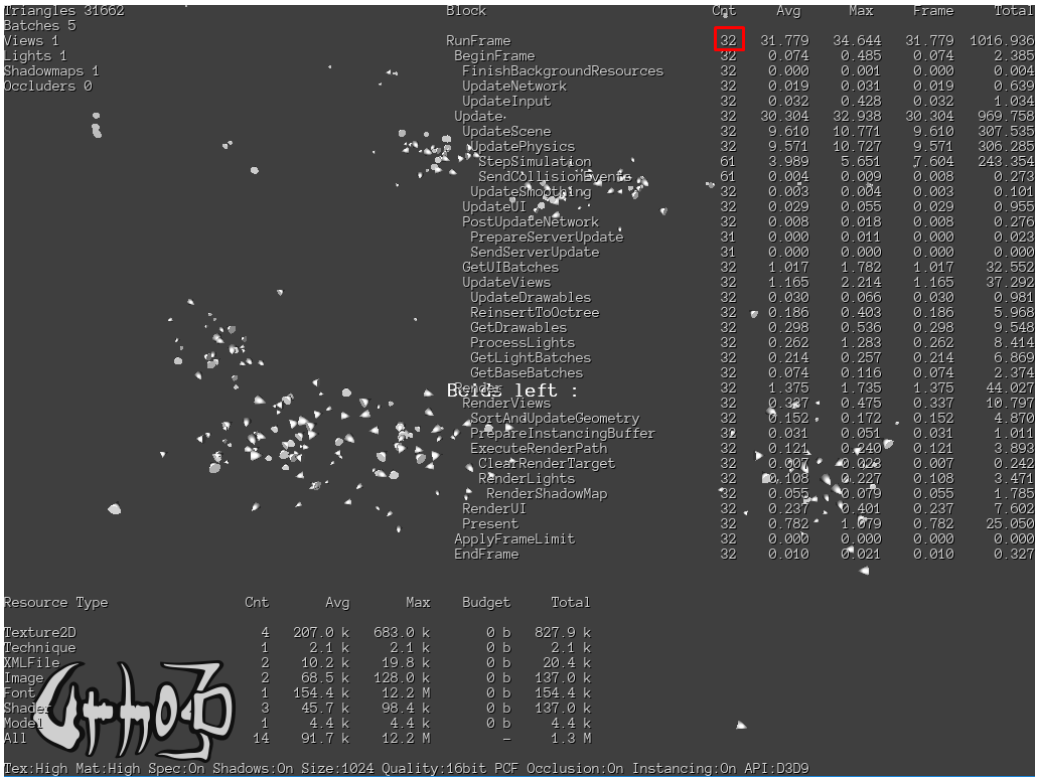


For another test, lets split this one big group of 300 into 2 smaller groups of 150 boids each, still equalling 300 boids altogether. The frame rate improves (24):



I am going to keep splitting the main group up to see how the frame rate will be affected.

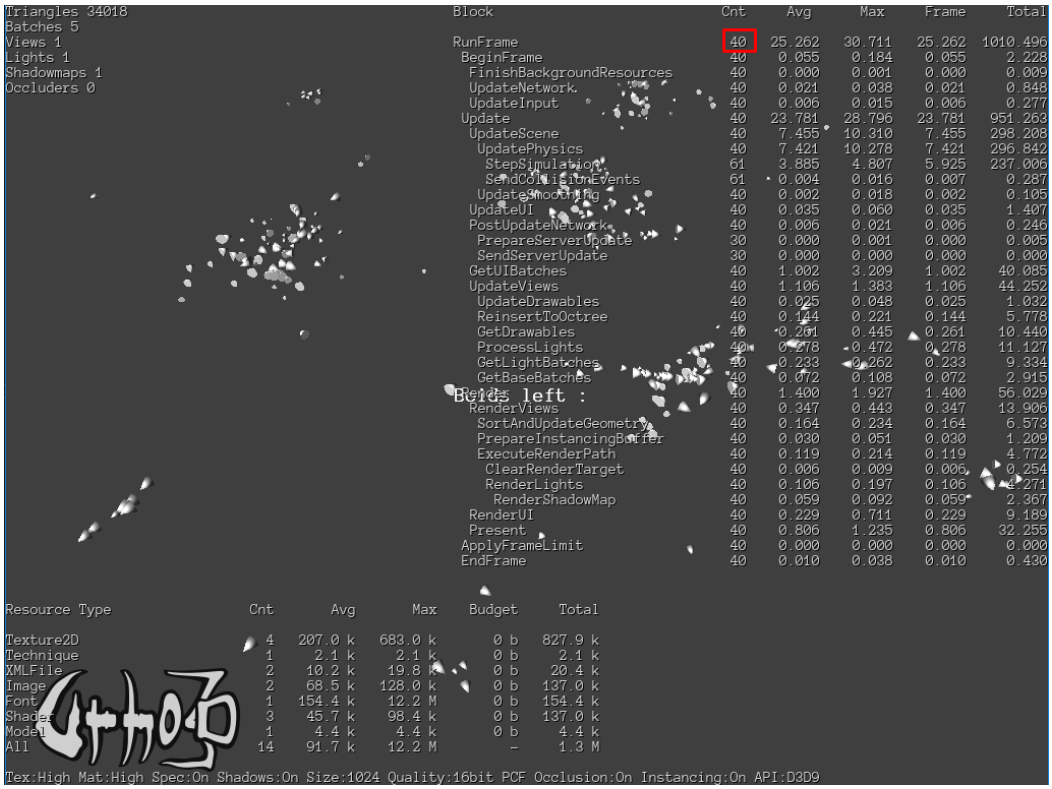
3 groups of boids, 100 in each. Frame rate (32) :



Block	Cnt	Avg	Max	Frame	Total
RunFrame	32	31.779	34.644	31.779	1016.936
BeginFrame	32	0.074	0.485	0.074	2.385
FinishBackgroundResources	32	0.000	0.001	0.000	0.004
UpdateNetwork	32	0.019	0.031	0.019	0.639
UpdateInput	32	0.032	0.428	0.032	1.034
Update	32	30.304	32.938	30.304	969.758
UpdateScene	32	9.610	10.771	9.610	307.535
UpdatePhysics	32	9.571	10.727	9.571	306.285
StepSimulation	61	3.989	5.651	7.604	243.354
SendCollisionEvents	61	0.004	0.009	0.008	0.273
UpdateSmoothing	32	0.003	0.004	0.003	0.101
UpdateUI	32	0.029	0.055	0.029	0.955
PostUpdateNetwork	32	0.008	0.018	0.008	0.276
PrepareServerUpdate	31	0.000	0.011	0.000	0.023
SendServerUpdate	31	0.000	0.000	0.000	0.000
GetUIBatches	32	1.017	1.782	1.017	32.552
UpdateViews	32	1.165	2.214	1.165	37.292
UpdateDrawables	32	0.030	0.066	0.030	0.981
ReinsertToOctree	32	0.186	0.403	0.186	5.968
GetDrawables	32	0.298	0.536	0.298	9.548
ProcessLights	32	0.262	1.283	0.262	8.414
GetLightBatches	32	0.214	0.257	0.214	6.869
GetBaseBatches	32	0.074	0.116	0.074	2.374
Boids	32	1.375	1.735	1.375	44.027
RenderViews	32	0.337	0.475	0.337	10.797
SortAndUpdateGeometry	32	0.152	0.172	0.152	4.870
PrepareInstancingBuffer	32	0.031	0.051	0.031	1.011
ExecuteRenderPath	32	0.121	0.240	0.121	3.893
ClearRenderTarget	32	0.007	0.028	0.007	0.242
RenderLights	32	0.108	0.227	0.108	3.471
RenderShadowMap	32	0.055	0.079	0.055	1.785
RenderUI	32	0.237	0.401	0.237	7.602
Present	32	0.782	1.079	0.782	25.050
ApplyFrameLimit	32	0.000	0.000	0.000	0.000
EndFrame	32	0.010	0.021	0.010	0.327

Resource Type	Cnt	Avg	Max	Budget	Total
Texture2D	4	207.0 k	683.0 k	0 b	827.9 k
Technique	1	2.1 k	2.1 k	0 b	2.1 k
XMLFile	2	10.2 k	19.8 k	0 b	20.4 k
Image	2	68.5 k	128.0 k	0 b	137.0 k
Font	1	154.4 k	12.2 M	0 b	154.4 k
Shader	3	45.7 k	98.4 k	0 b	137.0 k
Model	1	4.4 k	4.4 k	0 b	4.4 k
All	14	91.7 k	12.2 M	-	1.3 M

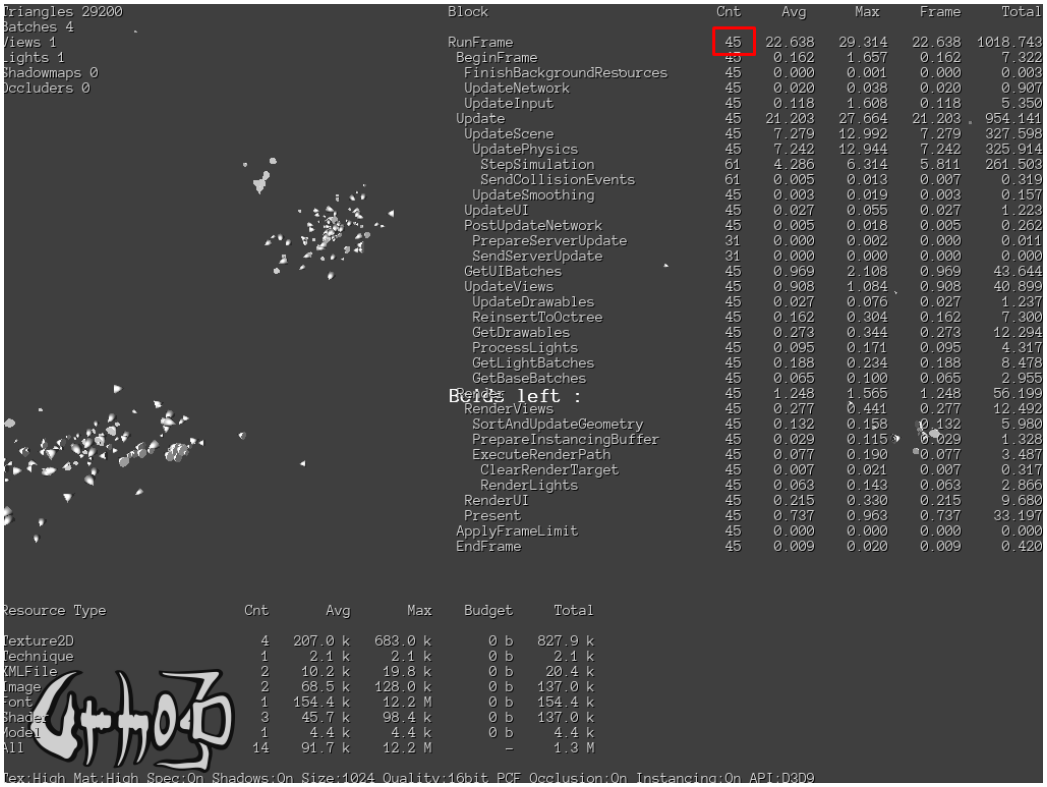
4 groups of boids, 75 in each. Frame rate (40) :



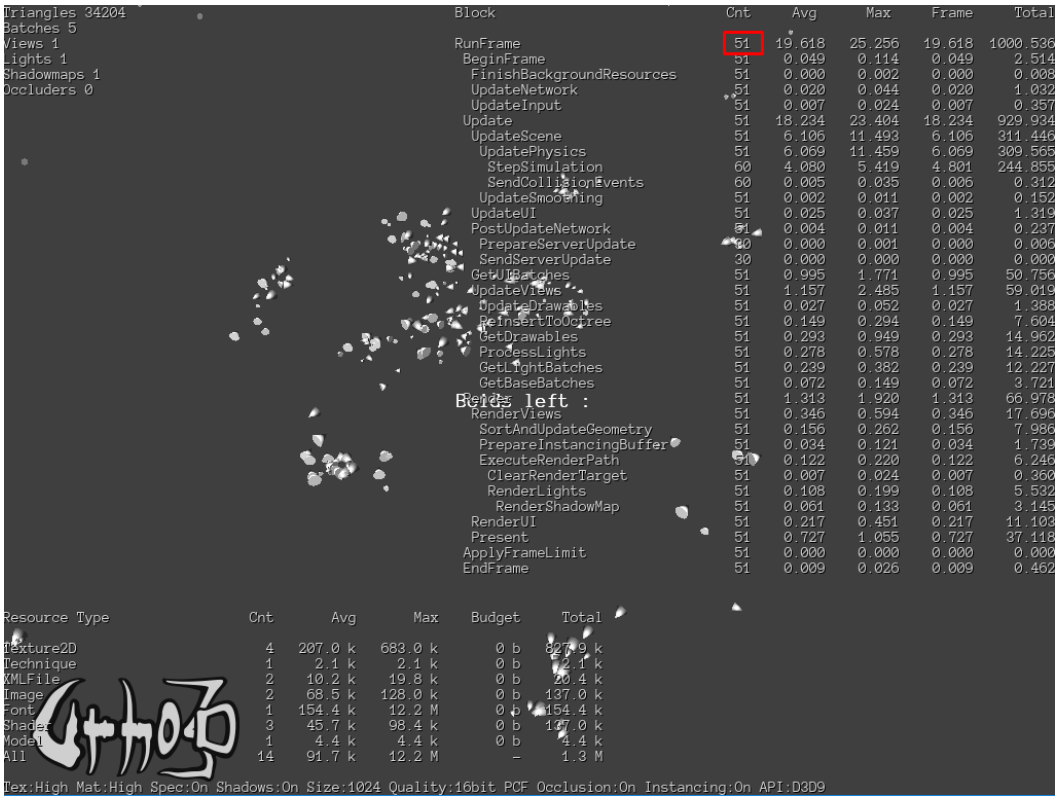
Block	Cnt	Avg	Max	Frame	Total
RunFrame	40	25.262	30.711	25.262	1010.496
BeginFrame	40	0.055	0.184	0.055	2.228
FinishBackgroundResources	40	0.000	0.001	0.000	0.009
UpdateNetwork	40	0.021	0.038	0.021	0.848
UpdateInput	40	0.006	0.015	0.006	0.277
Update	40	23.781	28.796	23.781	951.263
UpdateScene	40	7.455	10.310	7.455	298.208
UpdatePhysics	40	7.421	10.278	7.421	296.842
StepSimulation	61	3.885	4.807	5.925	237.006
SendCollisionEvents	61	0.004	0.016	0.007	0.287
UpdateSmoothing	40	0.002	0.018	0.002	0.105
UpdateUI	40	0.035	0.060	0.035	1.407
PostUpdateNetwork	40	0.006	0.021	0.006	0.246
PrepareServerUpdate	30	0.000	0.001	0.000	0.005
SendServerUpdate	30	0.000	0.000	0.000	0.000
GetUIBatches	40	1.002	3.209	1.002	40.085
UpdateViews	40	1.106	1.383	1.106	44.252
UpdateDrawables	40	0.025	0.048	0.025	1.032
ReinsertToOctree	40	0.144	0.221	0.144	5.778
GetDrawables	40	0.261	0.445	0.261	10.440
ProcessLights	40	0.278	0.472	0.278	11.127
GetLightBatches	40	0.233	0.262	0.233	9.334
GetBaseBatches	40	0.072	0.108	0.072	2.915
Boids	40	1.400	1.927	1.400	56.029
RenderViews	40	0.347	0.443	0.347	13.906
SortAndUpdateGeometry	40	0.164	0.234	0.164	6.573
PrepareInstancingBuffer	40	0.030	0.051	0.030	1.209
ExecuteRenderPath	40	0.119	0.214	0.119	4.772
ClearRenderTarget	40	0.006	0.009	0.006	0.254
RenderLights	40	0.106	0.197	0.106	4.271
RenderShadowMap	40	0.059	0.092	0.059	2.367
RenderUI	40	0.229	0.711	0.229	9.189
Present	40	0.806	1.235	0.806	32.255
ApplyFrameLimit	40	0.000	0.000	0.000	0.000
EndFrame	40	0.010	0.038	0.010	0.430

Resource Type	Cnt	Avg	Max	Budget	Total
Texture2D	4	207.0 k	683.0 k	0 b	827.9 k
Technique	1	2.1 k	2.1 k	0 b	2.1 k
XMLFile	2	10.2 k	19.8 k	0 b	20.4 k
Image	2	68.5 k	128.0 k	0 b	137.0 k
Font	1	154.4 k	12.2 M	0 b	154.4 k
Shader	3	45.7 k	98.4 k	0 b	137.0 k
Model	1	4.4 k	4.4 k	0 b	4.4 k
All	14	91.7 k	12.2 M	-	1.3 M

5 groups of boids, 60 in each. Frame rate (45) :



6 groups of boids, 50 in each. Frame rate (51) :



During these tests, you can see the gradual increase of the frame rate. During the test, I allowed around 10 seconds before checking and taking the screenshot of the framerate, this was to give the boids time to interact with each other, and this would be the most intensive part of the game, so it is good to test it at its most strained time.

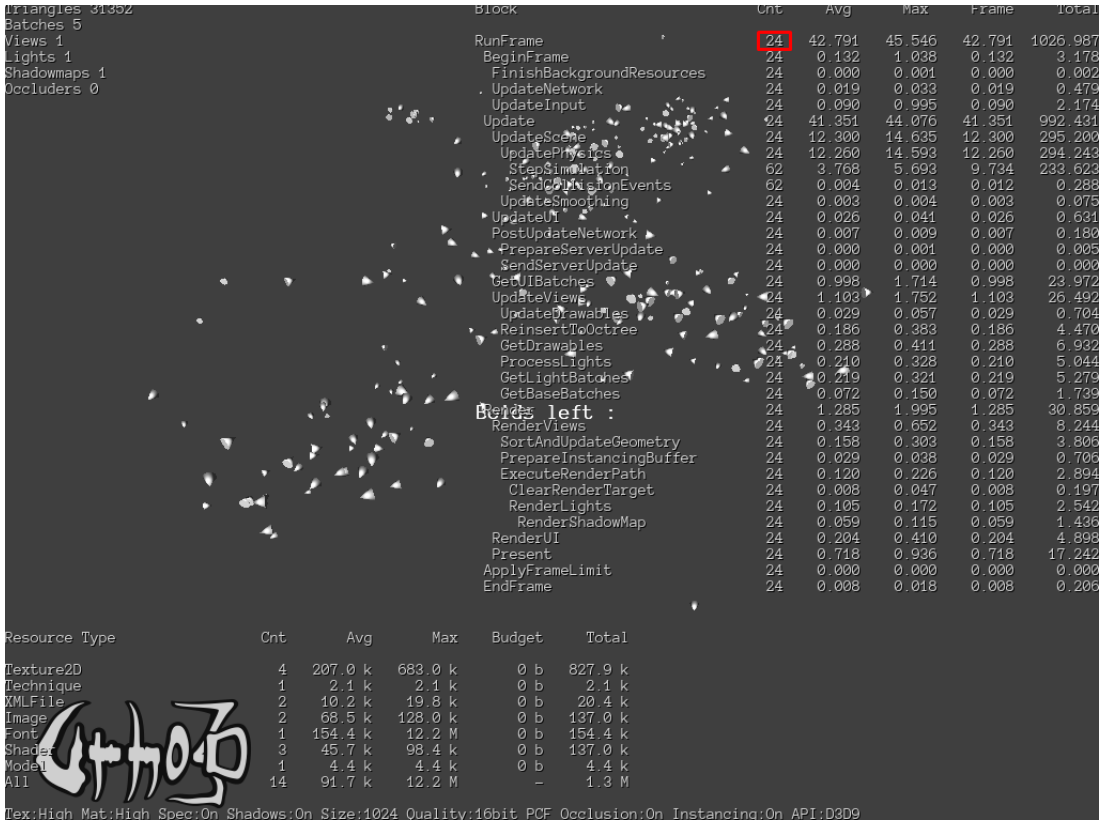
Boid splitting was a very good way to improve the frame rate of the game. Having 300 boids with the optimisation caused the frame rate to increase from 14 to 51. This is a very good result, but it must also be noted, that these individual boid groups do not interact with each other. Meaning, yes, we have a greatly improved frame rate, but the boids are actually interacting with the environment less. Meaning their behaviour will be slightly altered. Some boids will fly away not seeing its neighbours etc.

To conclude, this method is very good at improving the frame rate of the game, and it is very easy to implement, however, it can have a big effect on the boids behaviour.

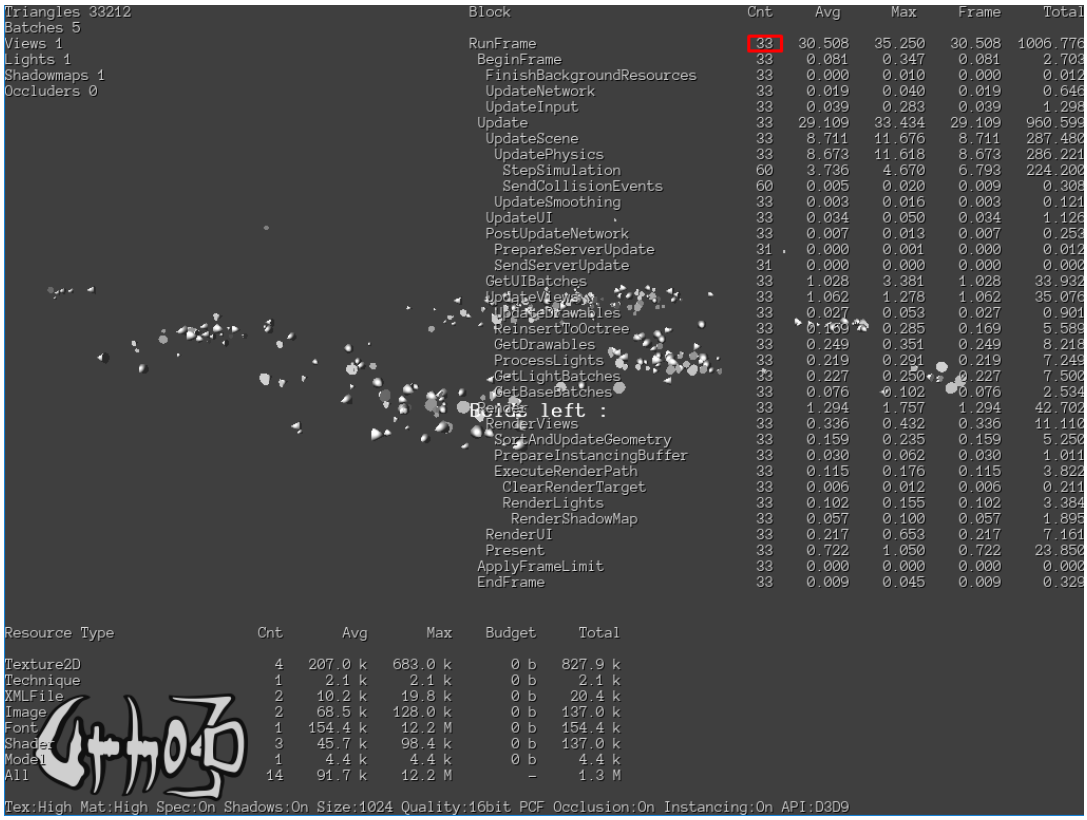
Limiting the Neighbours

In this method to try and improve optimisation, I am going to be counting the number of neighbours that each boid checks. I am going to get the total number of boids (300) and I am going to gradually make the game check for less and less neighbours to see how it affects the performance. The initial frame rate again, as seen in the above method, is 14. I have also made it, so it is only one group of boids, 300 in size, ignoring the previous method.

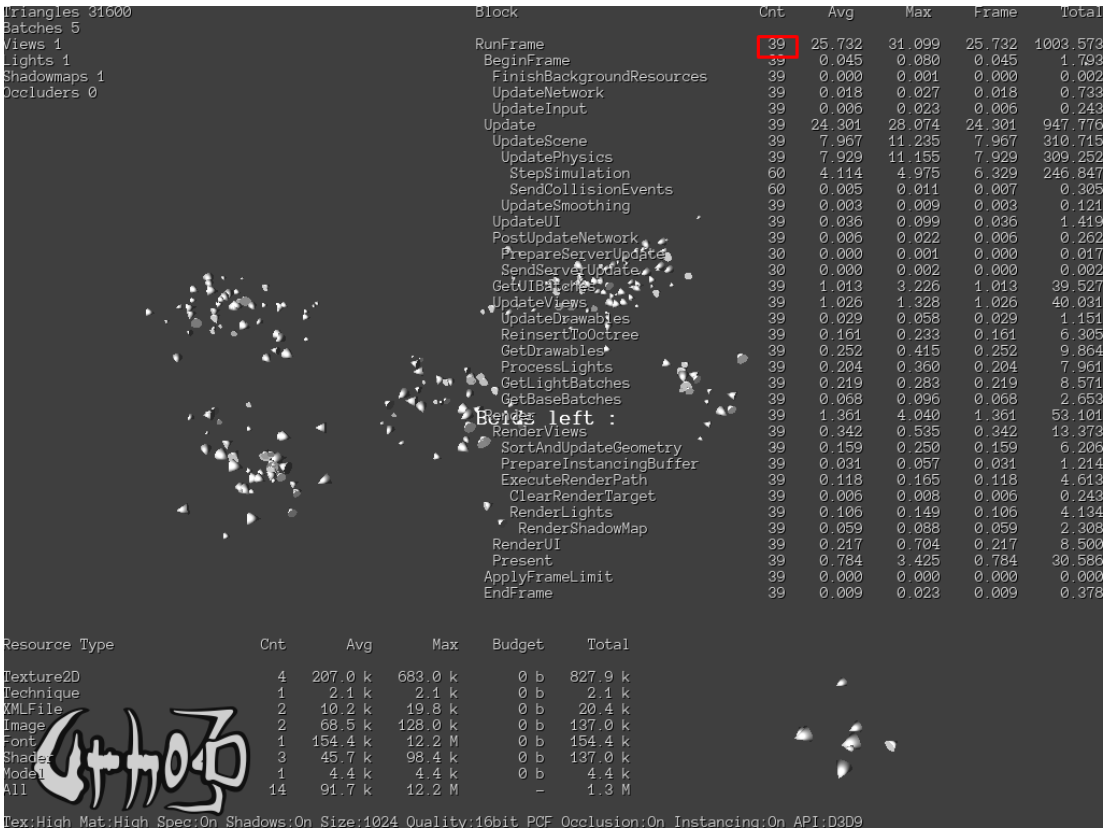
Checking for 1/2 of neighbours, frame rate (24) :



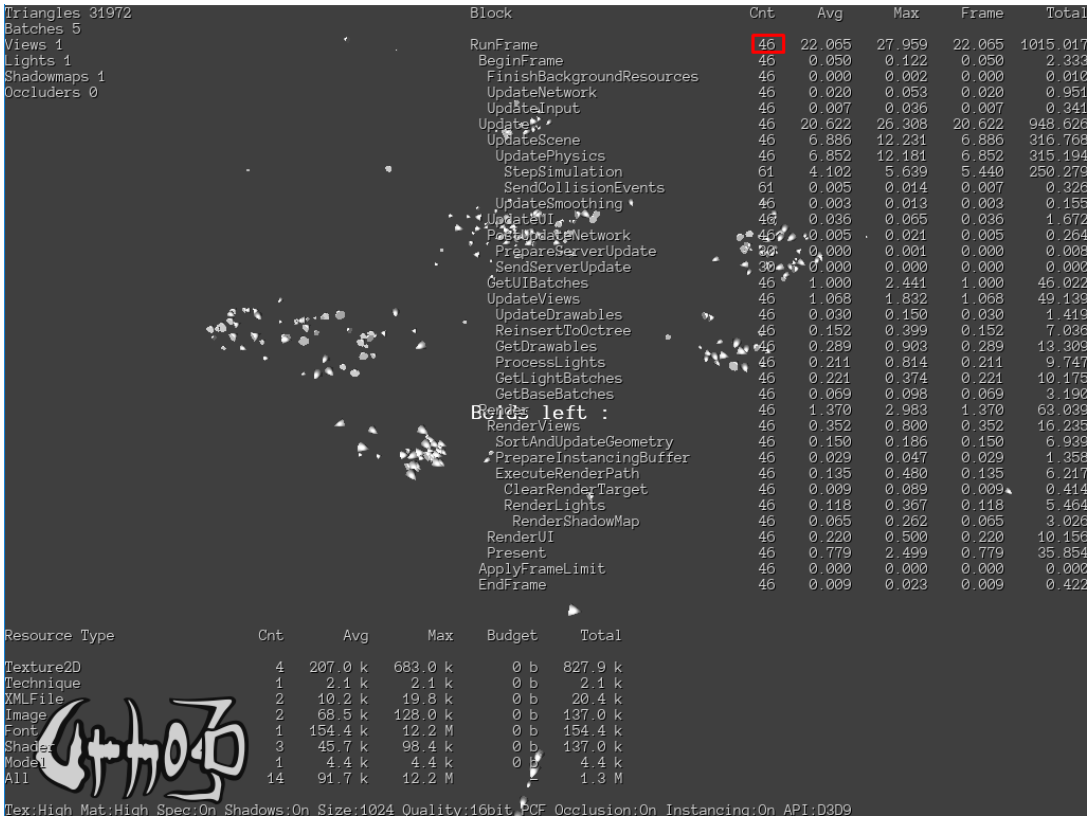
Checking for 1/3 of neighbours, frame rate (33) :



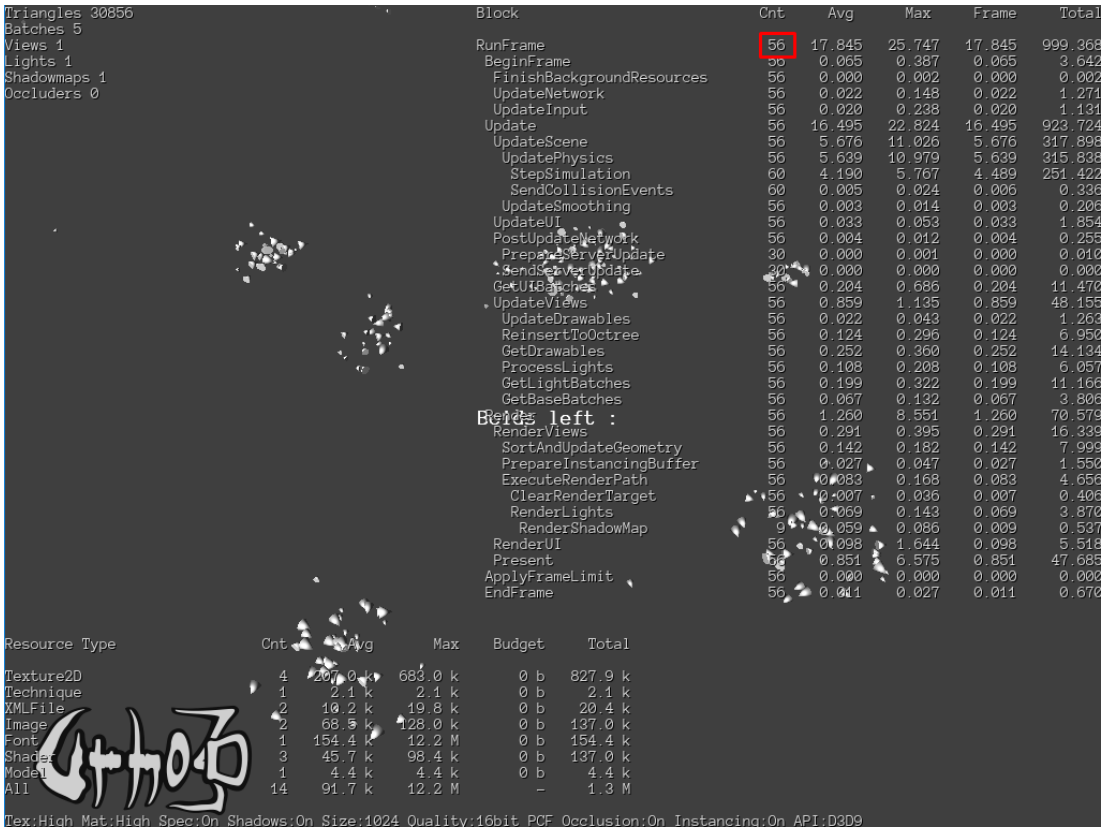
Checking for 1/4 of neighbours, frame rate (39) :



Checking for 1/5 of neighbours, frame rate (46) :



Checking for 1/6 of neighbours, frame rate (56) :

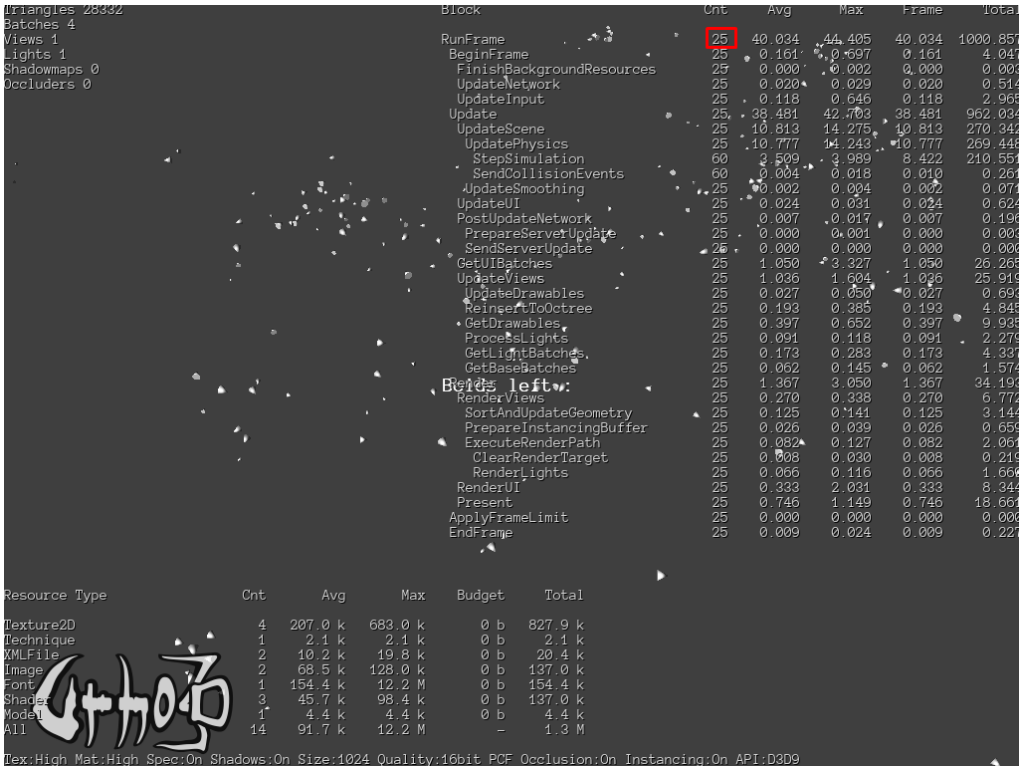


From these tests, once again, I saw an improvement in frame rate. As with the first method, as the frame rate gets better, this is because the boids are interacting with each other less. In the first method all of the boids interacting with each other in the same group, which is a positive, but the groups didn't interact. With this method, all of the boids are in the same group and are interactable, however, they cannot interact with all of the members in the group. This again will lose some of the behaviour of the boids and did mean that some of the boids did fly away alone, but you can tell a slight difference in the behaviours. In this method, I think they were better. As they can see the neighbours, and they're all in the same group, I think they stuck together more, and less flew away. Compared to the previous method were the group members might have been too far away from each other to interact, and therefore they got lost.

Boid Splitting

In this method, I only used one group of boids, 300 in size, and I updated the first half in one update loops, and then I updated the second half of the group in the next loops and repeated this.

Again, the initial frame rate is 14. When I used the method described about, the frame rate is (25) :



This method did show that the frame rate improves, however, it didn't show a big change on 300 boids. It also did affect the behaviour a little, as the boids were interacting less, they often got more spread from each other, and some even lost their neighbours completely. This was the worst method used.

Video

https://youtu.be/VFnq1CcB9dg