

## vertex manipulation in instanced shader hlsl SHARPDx 4.2

Asked 1 year, 9 months ago   Active 7 months ago   Viewed 159 times



0



[SOLVED] - The project is here [https://drive.google.com/open?id=1-kum0phM\\_iBMc8xRxYShmCNRWRs6W0Aw](https://drive.google.com/open?id=1-kum0phM_iBMc8xRxYShmCNRWRs6W0Aw)

I am using the same vertex shader and pixel shader for my instances. I am sending ONE FLAT array that contains all the instanced chunk bytes that make the cube of the Minecraft terrain. My goal is to make ONE DRAW CALL... Nothing is working. I am able to use an input element that has the specific index of the vertex logged in memory so all of the instances vertexes share the same index. It should be super easy after that to fetch the byte 0 or 1 inside of the flat chunk array that contains all of the data.

It should be working. The indexes of the vertices are perfect. The indexes of the instanced chunks are perfect...

What the h\*\*\* is NOT working? It feels as if something is wrong with the array as the chunks bytes are rendered randomly on my computer display. Everything that i see is wrong. I padded all of my structs to send to the shader and padded correctly the InputElements...??? I tried sending an array of Matrices instead... Each Matrix M11 to M44 contains the index of the bytes and I send that as an array to the shader and BAM... nothing is still working... Doing it on the cpu would be completely fine as i've done it a thousand times already.



```
cbuffer MatrixBuffer :register(b0)
{
    float4x4 world;
    float4x4 view;
    float4x4 proj;
}
```

By using our site, you acknowledge that you have read and understand our [Cookie Policy](#), [Privacy Policy](#), and [our Terms of Service](#).



```

    int chunkMap[4096];
}

struct VertexInputType
{
    float4 position : POSITION;
    float4 indexPos : POSITION1;
    float4 instancePosition1 : INSTANCEPOSITION1;
};

struct PixelInputType
{
    float4 position : SV_POSITION;
    float4 color : COLOR;
};

float planeSize = 1;

static int mapWidth = 4;
static int mapHeight = 4;
static int mapDepth = 4;

static int tinyChunkWidth = 4;
static int tinyChunkHeight = 4;
static int tinyChunkDepth = 4;

bool IsTransparent(int xMain, int yMain, int zMain, int x, int y, int z)
{
    int indexMain = xMain + mapWidth * (yMain + mapHeight * zMain);
    int indexSec = x + tinyChunkWidth * (y + tinyChunkHeight * z);

    if ((x < 0) || (y < 0) || (z < 0) || (x >= tinyChunkWidth) || (y >=
tinyChunkHeight) || (z >= tinyChunkDepth)) return true;
    {
        return chunkMap[(indexMain * 64) + indexSec] == 0;
    }
}

PixelInputType TextureVertexShader(VertexInputType input)
{
    PixelInputType output;

    input.position.w = 1.0f;

    int xMain = 0;
    int yMain = 0;
    int zMain = 0;

    xMain = round(fmod((input.instancePosition1.x*10), mapWidth*tinyChunkWidth));
    yMain = round(fmod((input.instancePosition1.y*10), mapHeight*tinyChunkHeight));
    zMain = round(fmod((input.instancePosition1.z*10), mapDepth*tinyChunkDepth));

    xMain = round(xMain / mapWidth);
    yMain = round(yMain / mapHeight);
    zMain = round(zMain / mapDepth);

    int x = input.indexPos.x;
    int y = input.indexPos.y;
    int z = input.indexPos.z;

```

```

int currentByte = chunkMap[(indexMain * 64) + indexSec];

input.position.x += input.instancePosition1.x;
input.position.y += input.instancePosition1.y;
input.position.z += input.instancePosition1.z;

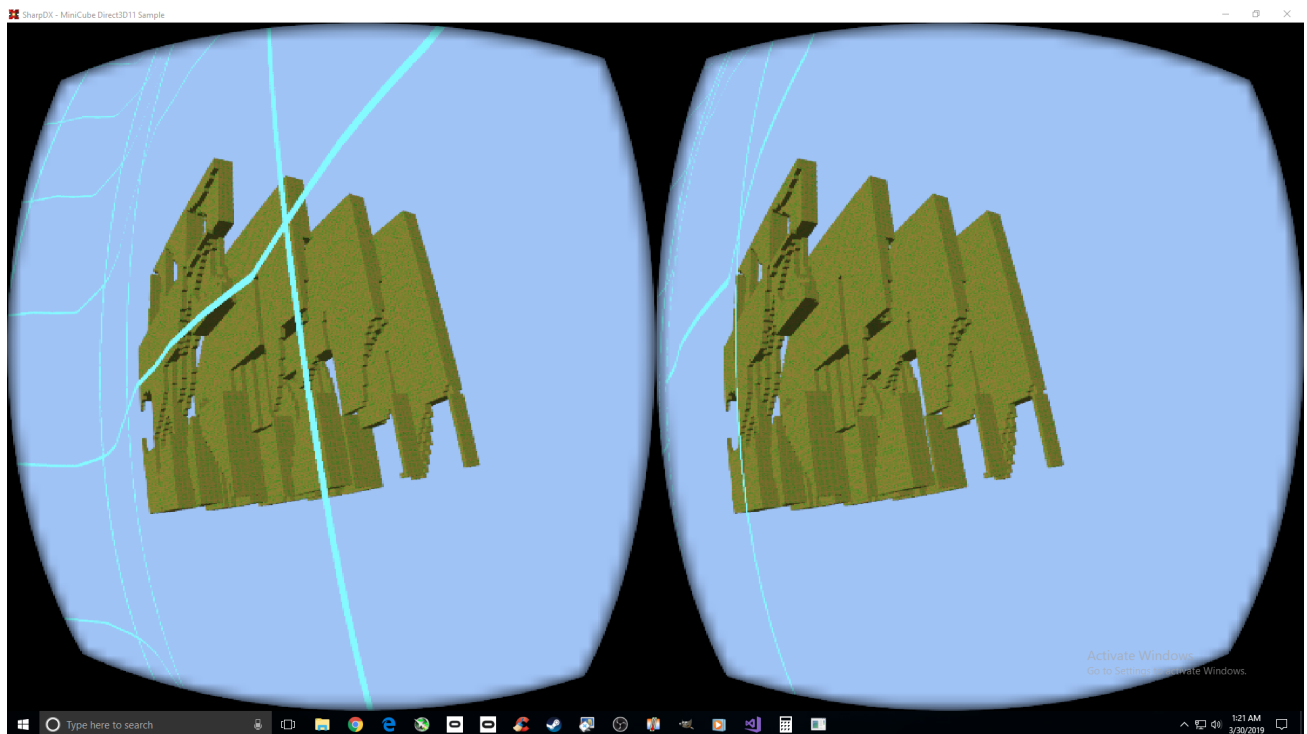
output.position = mul(input.position, world);
output.position = mul(output.position, view);
output.position = mul(output.position, proj);

if (currentByte == 1)
{
    output.color = float4(0,1,0,1);
    return output;
}
else
{
    output.color = float4(0,0,1,1);
    return output;
}
}

```

[Run code snippet](#)
[Copy snippet to answer](#)
[Expand snippet](#)

This is my current result. And it doesn't make sense and I don't understand why...



Seriously though. What is going on. I am sending a simple array to the GPU. Whatever padding I put shouldnt really matter since there is a single element in the struct for the buffer. Is it the GPU

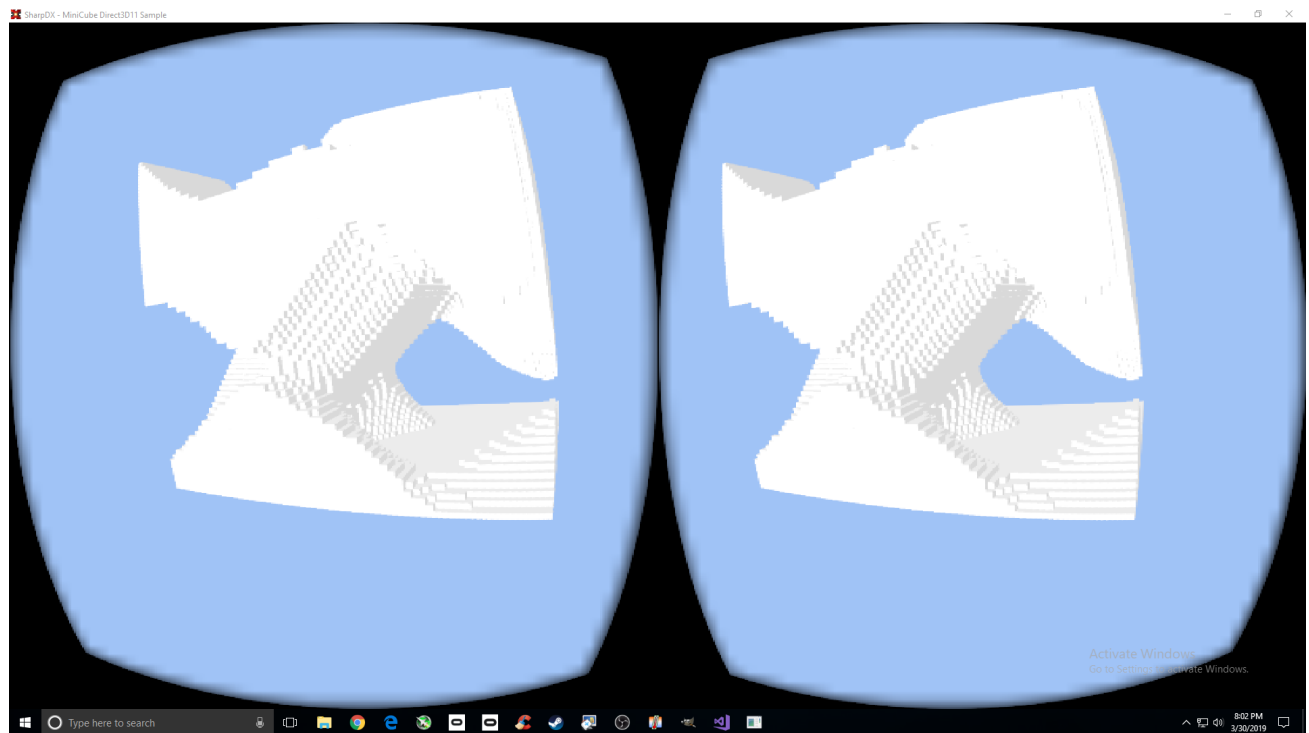
By using our site, you acknowledge that you have read and understand our [Cookie Policy](#), [Privacy Policy](#), and [our Terms of Service](#).



because I am setting those up BEFORE even starting rendering, they are setup as an input element Vector4 with x,y,z being the index position of the byte.

Or what I am trying is impossible? It shouldn't be unless it's related to some funky things the GPU is doing when instancing objects. Or I am just blind and I don't know what the f\*\*\* im coding.

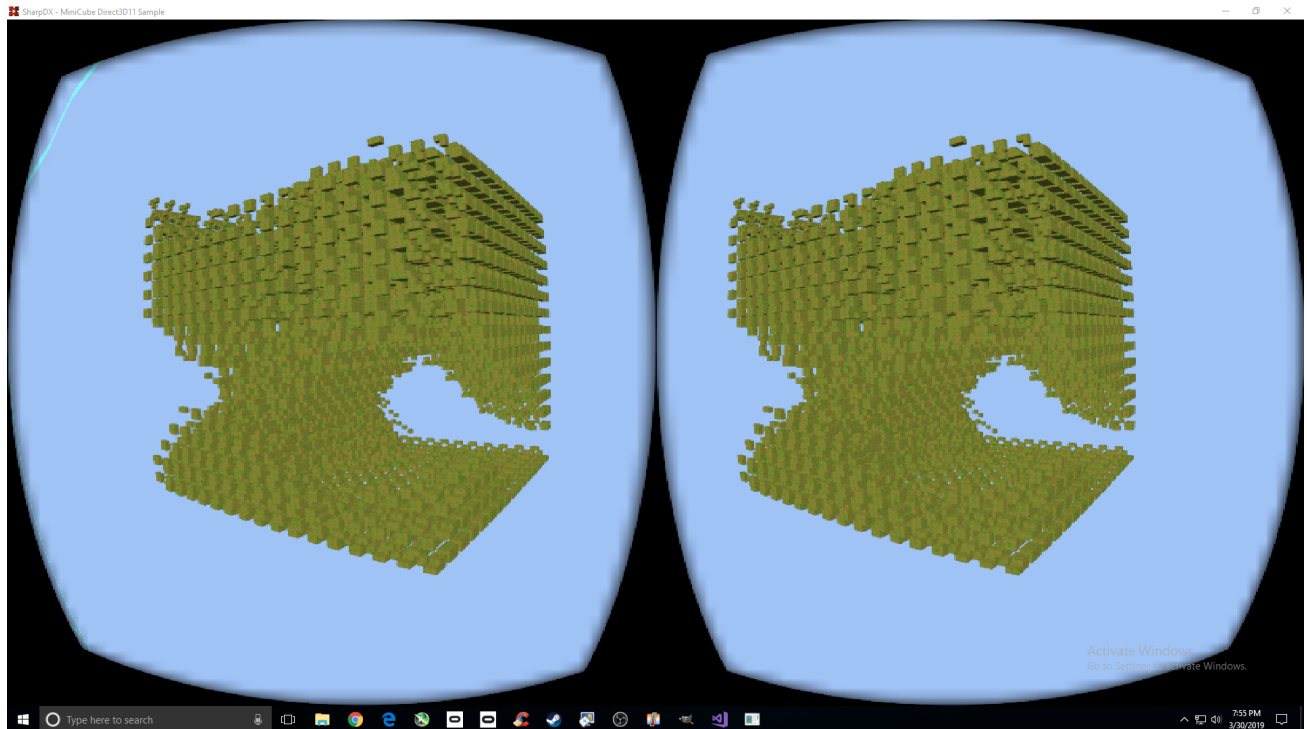
**EDIT 2019-03-30:** WHAT IT SHOULD LOOK LIKE (down below) - pic of chunk data that are drawn individually - more draw calls as each instance has its own draw calls...But this tiny picture shows instanced chunks with 4096 total instances. Each object created has 64 instances and there is 64 objects. There is 4096 draw calls in here at which point you can argue those aren't really instanced at all but they are! Each instance has its own draw call in order to modify the vertexes and indexes of each instances. It's a very bad programing test and no one should ever attempt this as it's completely useless... I mean, I could probably have better results even without instancing the chunks.



**EDIT 2019-03-30:** WHAT IT LOOKS RIGHT NOW (down below)- I am almost there but again it's not working yet. I am able to send chunk data another way instead of a constant buffer and it seems to do "half" of the job. I'm gonna keep at it. 1 draw call and i could make the terrain as big as I want it. Same thing with the chunks, 64 objects and 64 instances per object. BUT there is only 64 draw calls here hehe. Of course, I benchmarking at the same time and the final version won't be tiny chunksBytes of 4\*4\*4 by instances 4\*4\*4 by objects 4\*4\*4, they will be bigger.

As you can see, I got some good results finally where the chunks are correctly aligned and it seems that they are in the right position, but the chunks are missing bytes... I am wondering if it is a padding/packing issue because again, I went over the c# script many times to check the index

and 4 floats of 16 digits gives me 64 .... wow wait a minute. is this where I am doing it wrong? brb on this



nevermind, i am doing it right as I am building the VertexBufferBinding with an array of floats dismantled into a single float containing 16 digits for each slot. In the shader, i am accessing the vertexBinding and it really is giving me the correct float bytes, i think, for each instance. So everything should be ok. but its not ok.

**EDIT 2019-03-31: I have changed my shader to this instead.**

```
cbuffer MatrixBuffer :register(b0)
{
    float4x4 world;
    float4x4 view;
    float4x4 proj;
};

struct VertexInputType
{
    float4 position : POSITION;
    float4 indexPos : POSITION1;
    float4 color : COLOR;
    float3 normal : NORMAL;
    float2 tex : TEXCOORD;
    float3 dummyPad : NORMAL1;
    float4 indexPosMain : INSTANCEPOSITION;
    float4 instancePosition1 : INSTANCEPOSITION1;
};

struct PixelInputType
```

```

float2 tex : TEXCOORD;
};

float planeSize = 0.1f;

static int mapWidth = 4;
static int mapHeight = 4;
static int mapDepth = 4;

static int tinyChunkWidth = 4;
static int tinyChunkHeight = 4;
static int tinyChunkDepth = 4;

//[maxvertexcount(96)]
PixelInputType TextureVertexShader(VertexInputType input)
{
    PixelInputType output;
    input.position.w = 1.0f;

    //int mainX = input.indexPosMain.x;
    //int mainY = input.indexPosMain.y;
    //int mainZ = input.indexPosMain.z;

    int x = (input.indexPos.x);
    int y = (input.indexPos.y);
    int z = (input.indexPos.z);

    float currentMapData;
    float currentByte;

    int currentIndex = x + tinyChunkWidth * (y + tinyChunkHeight * z);

    if(currentIndex >= 0 && currentIndex <= 15)
    {
        currentMapData = input.indexPosMain.x;
        currentIndex = currentIndex;
        currentIndex = 15-currentIndex;
    }
    else if(currentIndex >= 16 && currentIndex <= 31)
    {
        currentMapData = input.indexPosMain.y;
        currentIndex = 31 - currentIndex;
        currentIndex = 15-currentIndex;
    }
    else if(currentIndex >= 32 && currentIndex <= 47)
    {
        currentMapData = input.indexPosMain.z;
        currentIndex = 47 - currentIndex;
        currentIndex = 15-currentIndex;
    }
    else if(currentIndex >= 48 && currentIndex <= 63)
    {
        currentMapData = input.indexPosMain.w;
        currentIndex = 63 - currentIndex;
        currentIndex = 15 - currentIndex;
    }

    if(currentIndex == 0)
    {
        float before = currentMapData * 0.1f;
        float newIndex1 = round(before);
    }
}

```

```

-----
{
    int tempData = currentMapData;
    for(int i = 0; i < 15-currentIndex; i++)
    {
        tempData = currentMapData * 0.1f;
        currentMapData = round(tempData);
    }

    float before = currentMapData * 0.1f;
    float newIndex1 = round(before);
    float lastIndex = before - newIndex1;
    float otherByte = lastIndex * 10;
    currentByte = round(otherByte);
}

int current = currentByte;

if(current == 1) //current >= 5 || wtf
{
    input.position.x += input.instancePosition1.x;
    input.position.y += input.instancePosition1.y;
    input.position.z += input.instancePosition1.z;

    output.position = mul(input.position, world);
    output.position = mul(output.position, view);
    output.position = mul(output.position, proj);
}
else
{
    input.position.x = input.instancePosition1.x;
    input.position.y = input.instancePosition1.y;
    input.position.z = input.instancePosition1.z;

    output.position = mul(input.position, world);
    output.position = mul(output.position, view);
    output.position = mul(output.position, proj);
}

output.color = input.color;
output.tex = input.tex;

output.normal = mul(input.normal, world);
output.normal = normalize(output.normal);

return output;
}

```

[Run code snippet](#)
[Copy snippet to answer](#)
[Expand snippet](#)

**EDIT: 2019-04-09 - 04h35am - I was able to make this work.**

c# sharpdx



Can you try editing your question to walk us through the strategy you're using here, and your understanding of how this *should* work? Right now a lot of this question is devoted to emphasizing how confused you are, which doesn't really help get a potential answerer up to speed on your technique. Maybe show a test case with a very small number of voxels, so you can show us the exact input data you're using, the expected output, and what output you get instead (annotated with axis labels) so we can try to work out what the pattern or relationship is. – [DMGregory](#) ♦ Mar 30 '19 at 8:17

I posted 2 pics showing what it should look like and what it looks like right now with my latest version. Its a benchmarking version so there is nothing fancy, just a chunk in the middle of nowhere with Perlin noise used. I do not have any axis setup yet and haven't used them for as long as I stopped coding in Unity and started working on Sharpdx. I will try to implement that soon but I use the same axis as in Unity for all of my projects. – [Ninekorn](#) Mar 31 '19 at 0:48

I have also posted my question in here to get some more attention. [gamedev.net/forums/topic/...](https://gamedev.net/forums/topic/...) – [Ninekorn](#) Mar 31 '19 at 1:46

I added my new shader version as I am now using one Vector4 that i send to the shader with only one vertex buffer binding. The Vector4 xyzw positions each contain a 16 digits that are made up of 0 and 1s. It looks like this in the Vector4: x:101001100111001011 and so on for the yzw . each axis of the vector has different data of course that are made of the chunks bytes 4\*4\*4. In the shader, I am also retrieving the index Byte of the vertex and calculating between 0 and 63 which index it is and then, i draw with a 1 and put the vertex somewhere else when its a 0. But i get those results up top – [Ninekorn](#) Apr 1 '19 at 2:51

Instead of stacking your question with more and more versions, I'd strongly recommend editing it to simplify down to a minimal example, with a detailed walkthrough of what you're doing now and why. The harder someone has to work to follow all the iterations of your process, the less likely they are to stick around long enough to give you detailed, useful answers. – [DMGregory](#) ♦ Apr 1 '19 at 4:07

If you've solved your problem, post your solution as an Answer that you can mark Accepted. Putting "(Solved)" in the title doesn't actually mark the issue as resolved within the StackExchange system. – [DMGregory](#) ♦ Apr 9 '19 at 9:17

I am worried about duplicata. I have built a Vertex Shader that Manipulates Vertex Position of Instanced object and it's not any of those shader iteration that I have posted here. I didn't use a constant buffer directly to manipulate the position of the vertexes. It all happens from vertex binding which is why it makes it so powerful. I didn't use I've haven't found anything like that yet anywhere. It took me two weeks and a half to build it. I will take a decision if I will share it or not. – [Ninekorn](#) Apr 9 '19 at 9:30


Already sharing what I have shared might lead people in the right path but my post doesn't seem that much interesting anyway. I mean I posted on Gamedev.net also and while getting over 400+ views, nobody helps. It feels like programing is a jungle where it's everyone for himself so I decided to close my post back there for the lack of help but the huge amount of views. it's not like I wanted to share my whole project but the shader in and of itself is the brick that holds the wall together. I am feeling bad for not sharing but I feel like beeing protective of this code for the moment. – [Ninekorn](#) Apr 9 '19 at 9:35


I think the reason you haven't gotten answers is that your question is very unclear, because you're sharing so little information about what you're doing. In my experience here, users love to help, when they can understand the problem clearly. Gifs of Leeloo don't help with that: detailed explanations and repro cases do. If you want others in the community to share knowledge and help with you, you need to be willing to share with them too. – [DMGregory](#) ♦ Apr 9 '19 at 9:39

The Leeloo pic was there to get people to connect with something, to feel that I really needed help just as she asks for help in The Fifth Element. Because I did need help. In fact, I've needed help for 2 years and a half and I have been 1000 times disappointed with how very little people are willing to share their




Apr 9 '19 at 10:07

1  This isn't about making someone happy, this is about leveraging the experience that's being offered to you. Longtime users of this site have helped refine and answer hundreds of questions each — they have good insights into what a question needs to facilitate useful answers. "Please help" ain't it. People are here because they already want to help, they just need to see an opportunity for how. If you've been asking for help the same way for two and a half years without results, while hundreds of other users are getting useful answers, maybe it's time to try some advice you've been offered. — [DMGregory](#) ♦ Apr 9 '19 at 10:56

I undeleted this post. I've always been the type of guy to share knowledge anyway. And when I try to do the opposite, I always feel bad doing so anyway. Here's what I am going to do. I will put a part of the project on Github with a license MIT so that everyone can see and I will complete this post to show the answer. Which means showing the complete shader... For its 177 lines... It's gonna be version 1 which is actually the simplest way that I've found to do this Vertex Shader Vertex Position Manipulator for procedural terrain. its coming soon. Its nothing fancy - just to show it works. — [Ninekorn](#) Apr 10 '19 at 1:34 

My project is here. [drive.google.com/open?id=1-kum0phM\\_iBMc8xRxYShmCNRWRs6W0Aw](https://drive.google.com/open?id=1-kum0phM_iBMc8xRxYShmCNRWRs6W0Aw) — [Ninekorn](#) Apr 15 '19 at 8:24

 Rather than editing questions to add "solved" to the title, you can accept an answer (which can be any answer, including your own) and the SE system will automatically handle this for you. — [Maximus Minimus](#) Apr 15 '19 at 8:59

Ok thank you. The answer is accepted. — [Ninekorn](#) Apr 15 '19 at 9:01

## 2 Answers

Active	Oldest	Votes
--------	--------	-------



1



The project Version 1.0 is here: [https://drive.google.com/open?id=1-kum0phM\\_iBMc8xRxYShmCNRWRs6W0Aw](https://drive.google.com/open?id=1-kum0phM_iBMc8xRxYShmCNRWRs6W0Aw)

The goal is achieved. It was to modify the vertex position inside of a Vertex Shader with using Vertex Binding Elements. The vertex binding is used both for storing the index of each byte (to which each vertex is assigned to) inside of each chunk instanced or not and also used to send the instance bytes in the form of integers.

In the end, the positions of each vertex is manipulated inside of the vertex shader but they are NOT removed from the scene. Hence why I will work on Version 1.1 and try to incorporate a Geometry shader to nullify triangles/faces that are not needed. In order to do that, I need to ask another question.

Oh... I forgot to mention one very important fact. In order to run the project, you need an Oculus Rift VR headset. Very soon, I will build the same project without the need of the VR headset. It does run faster without the headset as there is no need to draw twice for each eyes.

I didn't put a license on my shader and whatnot because I got bored searching on how to put the project on GitHub + I am the TUCO of coding, so every part of my code here and there uses parts of someone else's code so in the end. I didn't know what to put a license on.



edited Apr 15 '19 at 8:45

answered Apr 15 '19 at 8:31

Ninekorn



0



I am sorry, this thread is a huge mess. i am really late in updating my things and i am sorry for that.

I failed to provide the explanations as to why my chunk system was powerful.

The explanations that i had given and the shader that i had posted up above were not my final byteshift operator lock on a digit inside of the float where i am storing the byte position (to the left of the dot in 0.0 and to the right of that dot) which is an addition to the chunk system that Craig Perko built and that is available on youtube already.

I wanted to get attention right away but it didnt work. 1st version is a very powerful instanced chunk placement system for the position of an area chunk that contains the instances of the tiny chunks and those bytes inside of that chunk also are coming in the shader. With the use of byte shift operator to lock the position inside of a float, i am able to get those necessary "1s" and "0s" that decides if yes or no to build the 6 faces of that tiny cubes that is the mesh of a chunk that is an instance of the area chunk that contains those chunk instances.

rounding only wasnt working. flooring only wasnt working.

it's all due to finding this here: <https://stackoverflow.com/questions/46312893/how-do-you-use-bit-shift-operators-to-find-out-a-certain-digit-of-a-number-in-ba>

my 1stVersion is not a complicated example but it has absolutely no comments and some stupid variable names as i was tired after many attempts in that shader always rewriting new variable names or commenting the code out etc, to make my byte/float vertex binding system, instead i just add a "vowel" and then a consonant" or vice-versa, to move on forward but that was in september 2019. yes i name my variable names right this time around and i make sure to leave myself better hints. if in any case this has helped you but that you never admitted it was coming from someone else, you are not doing the right thing.

Shader:

```
cbuffer MatrixBuffer :register(b0)
{
    float4x4 world;
    float4x4 view;
    float4x4 proj;
};

/*cbuffer MatrixBuffer :register(b1)
```



```

struct VertexInputType
{
    float4 position : POSITION0;
    float4 indexPos : POSITION1;

    float4 color : COLOR0;
    float3 normal : NORMAL0;
    float2 tex : TEXCOORD0;
    int one : PSIZE0;
    int two : PSIZE1;
    int three : PSIZE2;
    int four : PSIZE3;
    int oneTwo : PSIZE4;
    int twoTwo : PSIZE5;
    int threeTwo : PSIZE6;
    int fourTwo : PSIZE7;
    float4 instancePosition1 : POSITION2;
};

//row_major matrix instancePosition1 : WORLD;

struct PixelInputType
{
    float4 position : SV_POSITION;
    float4 color : COLOR0;
    float3 normal : NORMAL0;
    float2 tex : TEXCOORD0;
};

//float planeSize = 0.1f;

static int mapWidth = 4;
static int mapHeight = 4;
static int mapDepth = 4;

static int tinyChunkWidth = 4;
static int tinyChunkHeight = 4;
static int tinyChunkDepth = 4;

//[maxvertexcount(96)]
PixelInputType TextureVertexShader(VertexInputType input)
{
    PixelInputType output;
    input.position.w = 1.0f;

    int x = input.indexPos.x;
    int y = input.indexPos.y;
    int z = input.indexPos.z;

    int currentMapData;
    int currentByte;

    int currentIndex = x + (tinyChunkWidth * (y+(tinyChunkHeight*z)));
    int someOtherIndex = currentIndex;

    int theNumber = tinyChunkWidth;
    int remainder = 0;
    int totalTimes = 0;

    for (int i = 0; i <= currentIndex; i++)
    {
        if (remainder == theNumber)

```

```

    if (totalTimes * theNumber >= currentIndex)//>=?? why not only >
    {
        break;
    }
    remainder++;
}

int arrayIndex = int(floor(totalTimes *0.5));

switch(arrayIndex)
{
    case 0:
        currentMapData = input.one;
        break;
        case 1:
            currentMapData = input.oneTwo;
            break;
    case 2:
        currentMapData = input.two;
        break;
    case 3:
        currentMapData = input.twoTwo;
        break;
    case 4:
        currentMapData = input.three;
        break;
    case 5:
        currentMapData = input.threeTwo;
        break;
    case 6:
        currentMapData = input.four;
        break;
    case 7:
        currentMapData = input.fourTwo;
        break;
}

//0-4-1-5-2-6-3-7
//8-12-9-13-10-14-11-15
//16-20-17-21-18-22-19-23
//24-28-25-29-26-30-27-31
//32-36-33-37-34-38-35-39
//40-44-41-45-42-46-43-47
//48-52-49-53-50-54-51-55
//56-60-57-61-58-62-59-63

int baser = totalTimes;

int someAdder = totalTimes % 2;

someOtherIndex=7-(((someOtherIndex-(tinyChunkWidth*baser))*2)+someAdder);

int testera = 0;
int subtract = 0;
int before0 = 0;

if (someOtherIndex == 0)
{
    testera = currentMapData >> 1 << 1;
    currentByte = currentMapData - testera;
}

```

```

for (int i = 0; i < someOtherIndex; i++)
{
    someData0 = int(someData0 * 0.1f);
}

before0 = int(trunc(someData0));
//https://stackoverflow.com/questions/46312893/how-do-you-use-bit-shift-
operators-to-find-out-a-certain-digit-of-a-number-in-ba
testera = before0 >> 1 << 1;
currentByte = before0 - testera;
}

//currentByte = 1;

if(currentByte == 1)
{
    input.position.x += input.instancePosition1.x;
    input.position.y += input.instancePosition1.y;
    input.position.z += input.instancePosition1.z;

    output.position = mul(input.position, world);
    output.position = mul(output.position, view);
    output.position = mul(output.position, proj);
    output.color = input.color;
}
else
{
    input.position.x = input.instancePosition1.x;
    input.position.y = input.instancePosition1.y;
    input.position.z = input.instancePosition1.z;

    output.position = mul(input.position, world);
    output.position = mul(output.position, view);
    output.position = mul(output.position, proj);
    output.color = input.color * float4(0.5f,0.5f,0.5f,1);
}

output.tex = input.tex;

output.normal = mul(input.normal, world);
output.normal = normalize(output.normal);

return output;
}

/*technique Test
{
    pass pass0 //pass1
    {
        VertexShader = compile vs_5_0 TextureVertexShader();
        //PixelShader = compile ps_5_0 TexturePixelShader();
    }
}*/

```

The link of the official version is still there and i am terribly sorry that i didn't review it as i think i can build much much better now.

If it "inspired" you to move on with your projects, if my "template/blueprint" of "how it's made/how it works" helped you build your own scripts just say a thank you, it would make my day and you

edited May 9 at 7:39

answered May 8 at 16:27

 **Ninekorn**

11

5

