Introduction to Graphics Programming and its Applications

繪圖程式設計與應用

Assignment 3
GPU Driven Rendering

Instructor: Hung-Kuo Chu

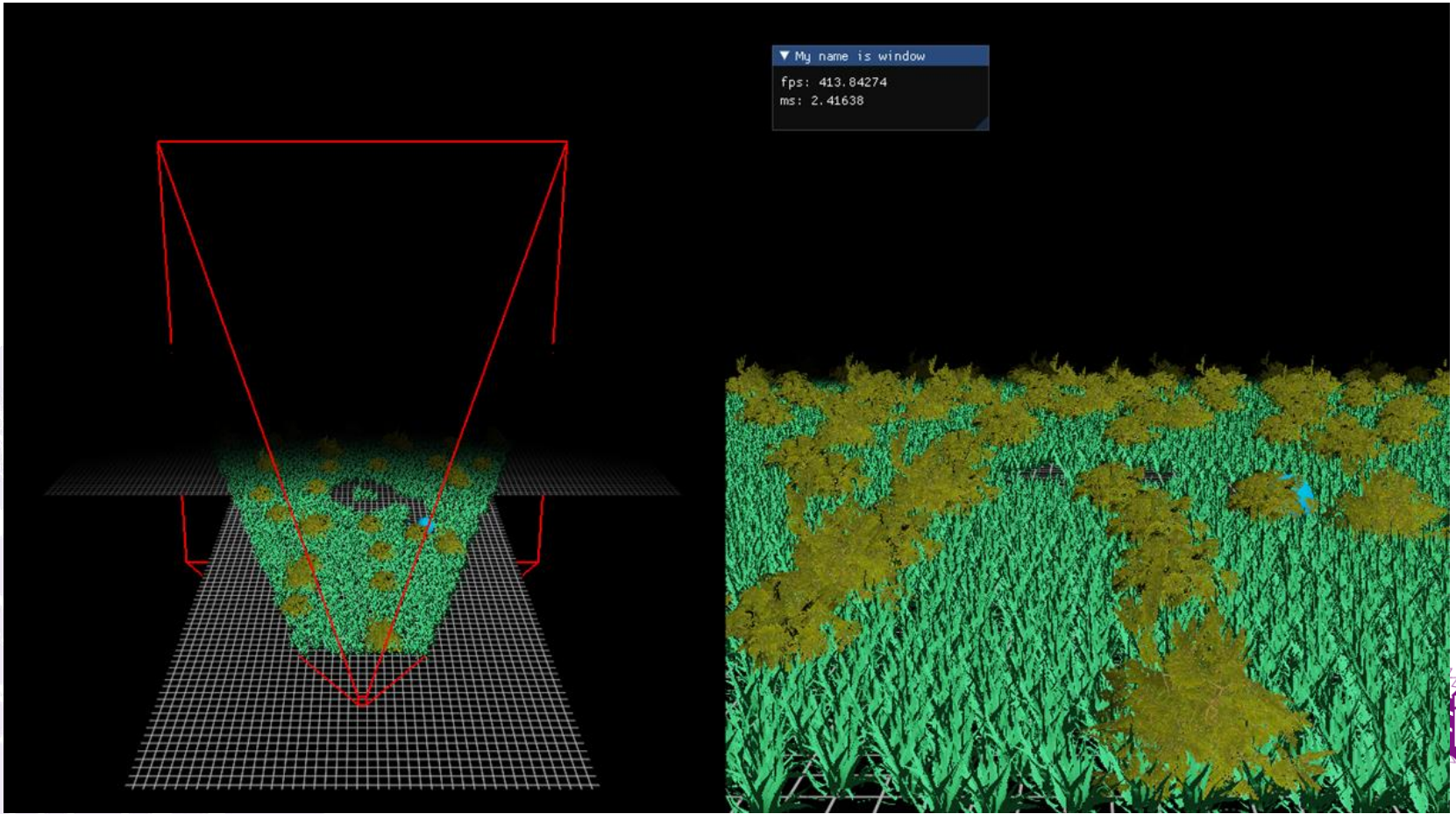
Department of Computer Science

National Tsing Hua University

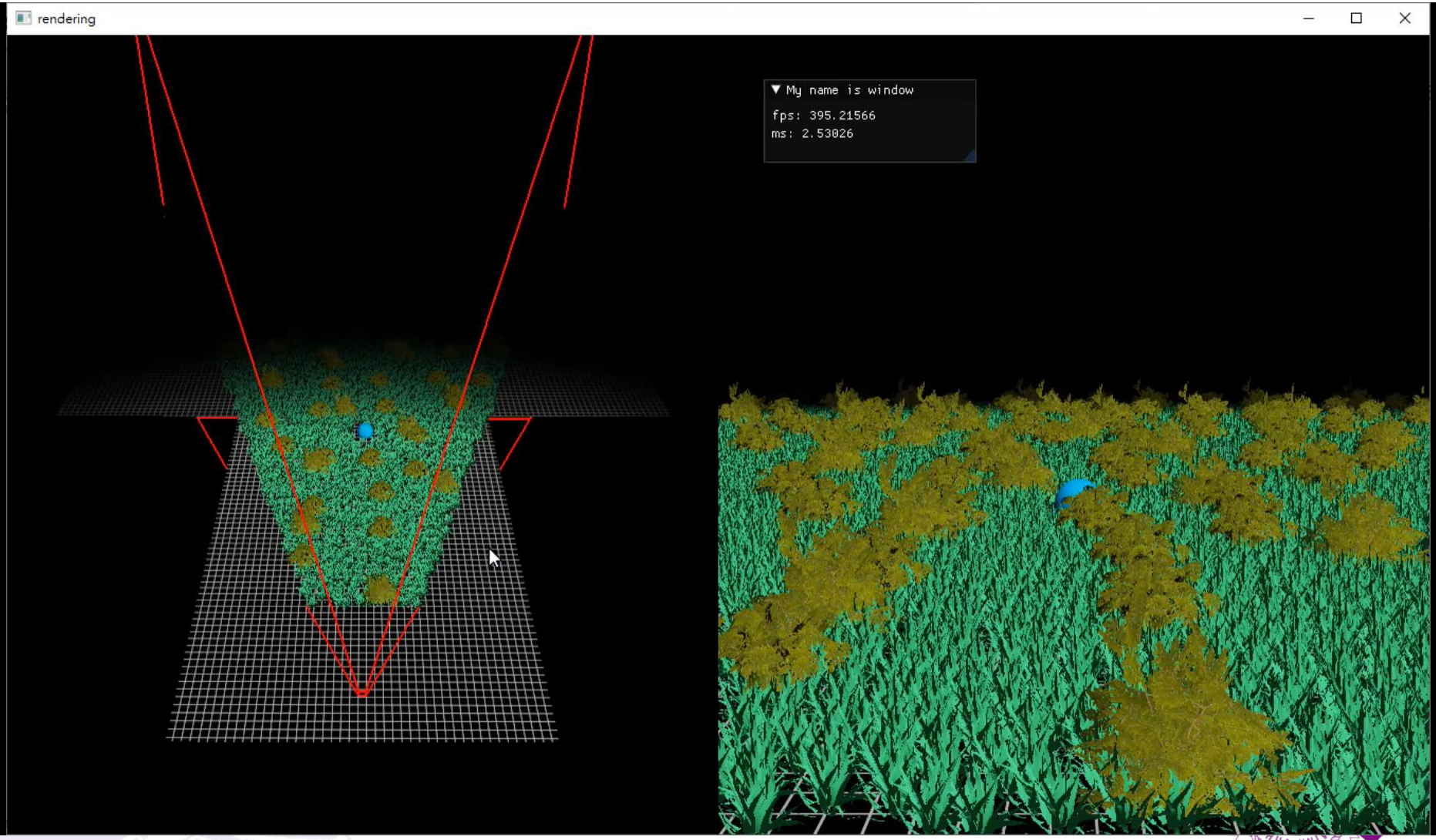
CS5507

Introduction

- Use **(multi) indirect draw** technique to render the scene
- Use **compute shader** to dynamically update instance buffers and draw commands
  - View frustum culling, instance erasing



Demo Video



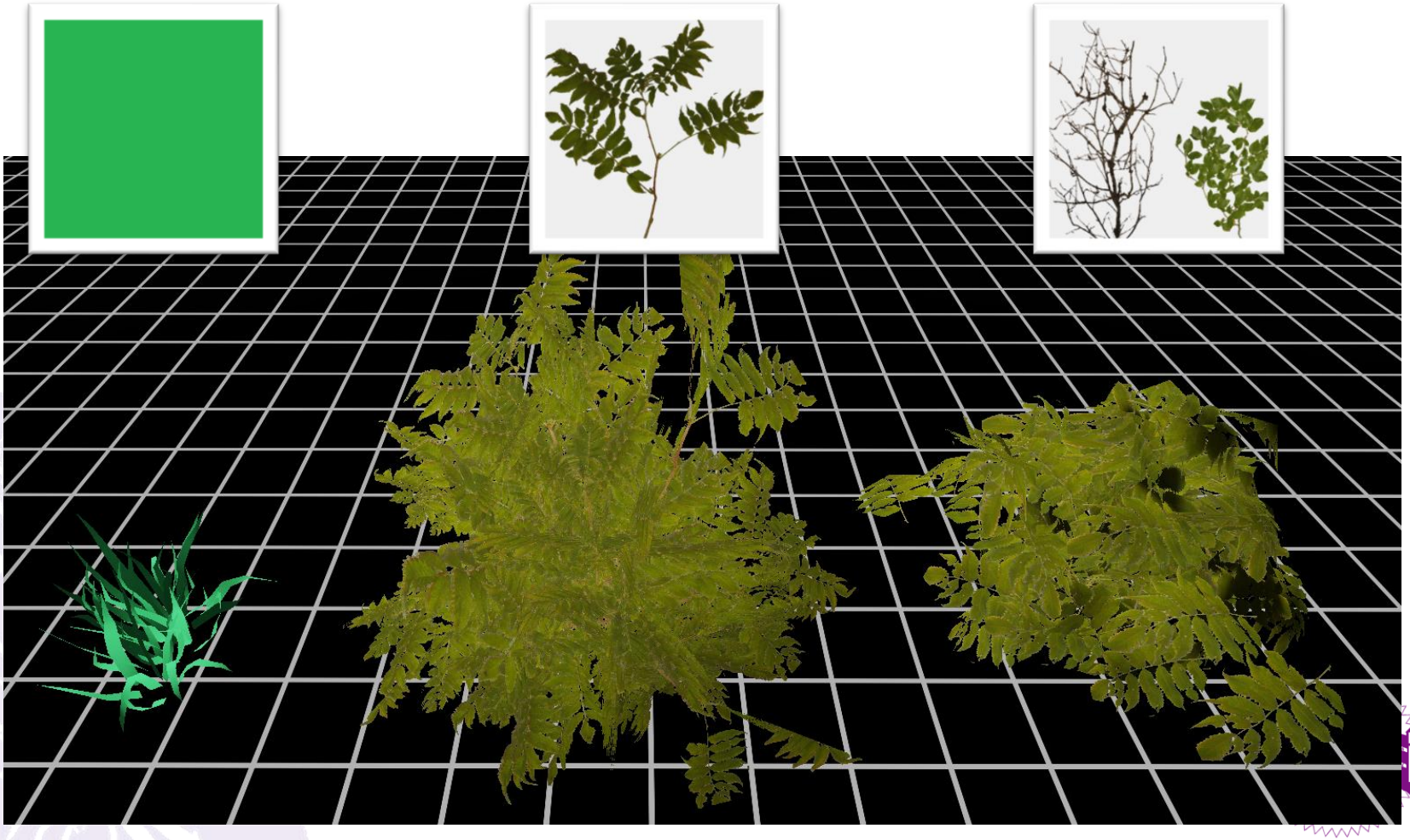
SSBO Definition Example

```
struct DrawCommand{
  uint count ;
  uint instanceCount ;
  uint firstIndex ;
  uint baseVertex ;
  uint baseInstance ;
};

/* the SSBO for storing draw commands */
layout (std430, binding=/*determine by yourself*/) buffer DrawCommandsBlock{
  DrawCommand commands[] ;
};

struct RawInstanceProperties{
  vec4 position ;
  ivec4 indices ;
};

struct InstanceProperties{
  vec4 position ;
};

/*the buffer for storing “whole” instance position and other necessary information*/
layout (std430, binding=/*determine by yourself*/) buffer RawInstanceData
{
  RawInstanceProperties rawInstanceProps[] ;
};

/*the buffer for storing “visible” instance position*/
layout (std430, binding=/*determine by yourself*/) buffer CurrValidInstanceData
{
  InstanceProperties currValidInstanceProps[] ;
};
```

Texture 2D Array

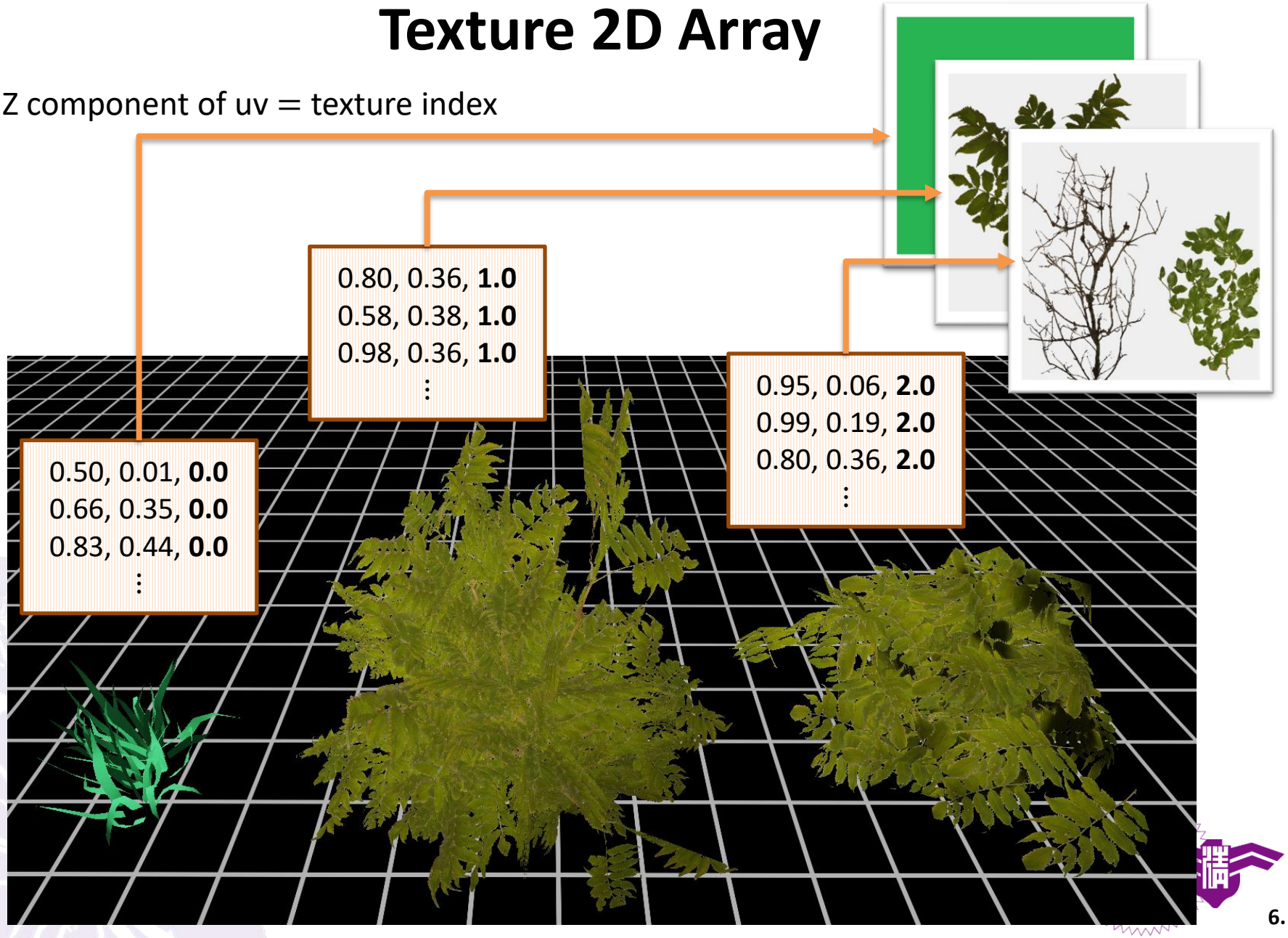
Question:
How to tell shader to use correct texture for each mesh ?

- There are different textures for different meshes



Texture 2D Array

Z component of uv = texture index



Texture 2D Array

Note: There is no GL_TEXTURE_3D_ARRAY !

```
const int NUM_TEXTURE = 3;
const int IMG_WIDTH = 1024;
const int IMG_HEIGHT = 1024;
const int IMG_CHANNEL = 4 ;

uchar* textureArrayData = new uchar*[IMG_WIDTH * IMG_HEIGHT * IMG_CHANNEL * NUM_TEXTURE] ;
// merge the textures to the texture array data
…

// create texture array
glGenTextures(1, &textureArrayHandle);
glBindTexture(GL_TEXTURE_2D_ARRAY, textureArrayHandle);

// the internal format for glTexStorageXD must be "Sized Internal Formats“
// max mipmap level = log2(1024) + 1 = 11
glTexStorage3D(GL_TEXTURE_2D_ARRAY, 11, GL_RGBA8, IMG_WIDTH, IMG_HEIGHT, NUM_TEXTURE);
glTexSubImage3D(GL_TEXTURE_2D_ARRAY, 0, 0, 0, 0, IMG_WIDTH, IMG_HEIGHT, NUM_TEXTURE, GL_RGBA,
GL_UNSIGNED_BYTE, textureArrayData);

glTexParameterf(GL_TEXTURE_2D_ARRAY, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
glTexParameterf(GL_TEXTURE_2D_ARRAY, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
glTexParameterf(GL_TEXTURE_2D_ARRAY, GL_TEXTURE_WRAP_S, GL_REPEAT);
glTexParameterf(GL_TEXTURE_2D_ARRAY, GL_TEXTURE_WRAP_T, GL_REPEAT);

// IMPORTANT !! Use mipmap for the foliage rendering
glGenerateMipmap(GL_TEXTURE_2D_ARRAY)
```

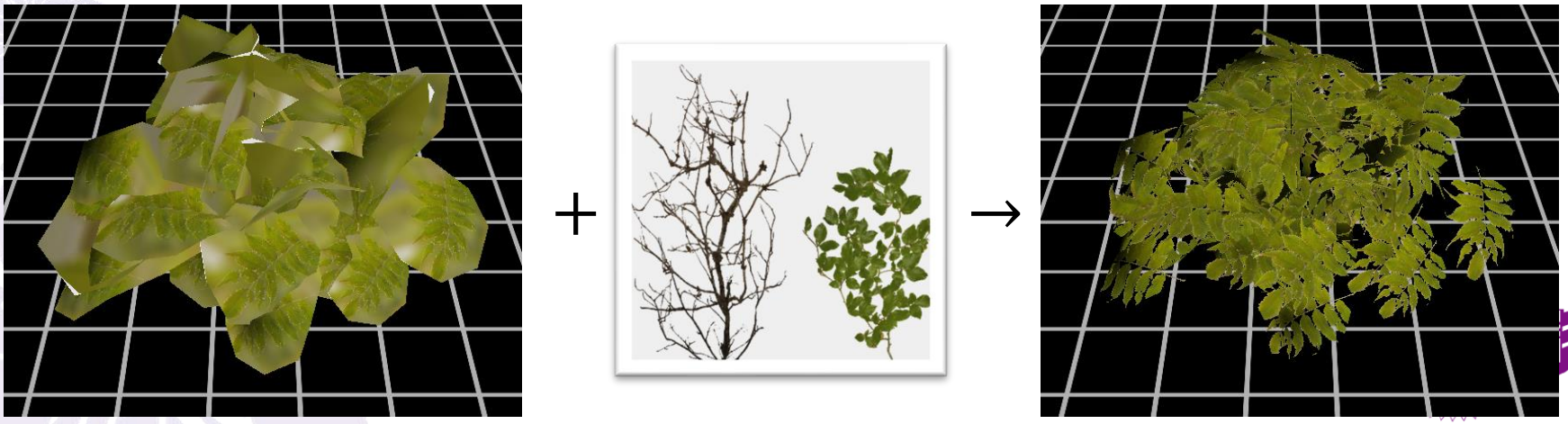
Texture 2D Array

```
#version 430 core

// the input from rasterizer
in vec3 f_uv ;
out vec4 fragColor ;
uniform sampler2DArray albedoTextureArray ;

void main(){
  // once the sampler is sampler2DArray, the uv must be vec3
  vec4 texel = texture(albedoTextureArray, f_uv) ;

  // discard the transparent texel
  if(texel.a < 0.5){
    discard ;
  }

  // output color
  fragColor = texel ;
}
```

Transparent Texel Discarding

```
#version 430 core

// the input from rasterizer
…

void main(){
  vec4 texel = texture(albedoTextureArray, f_uv) ;

  // discard the transparent texel
  if(texel.a < 0.5){
    discard ;
  }
  // output color
  fragColor = texel ;
}
```

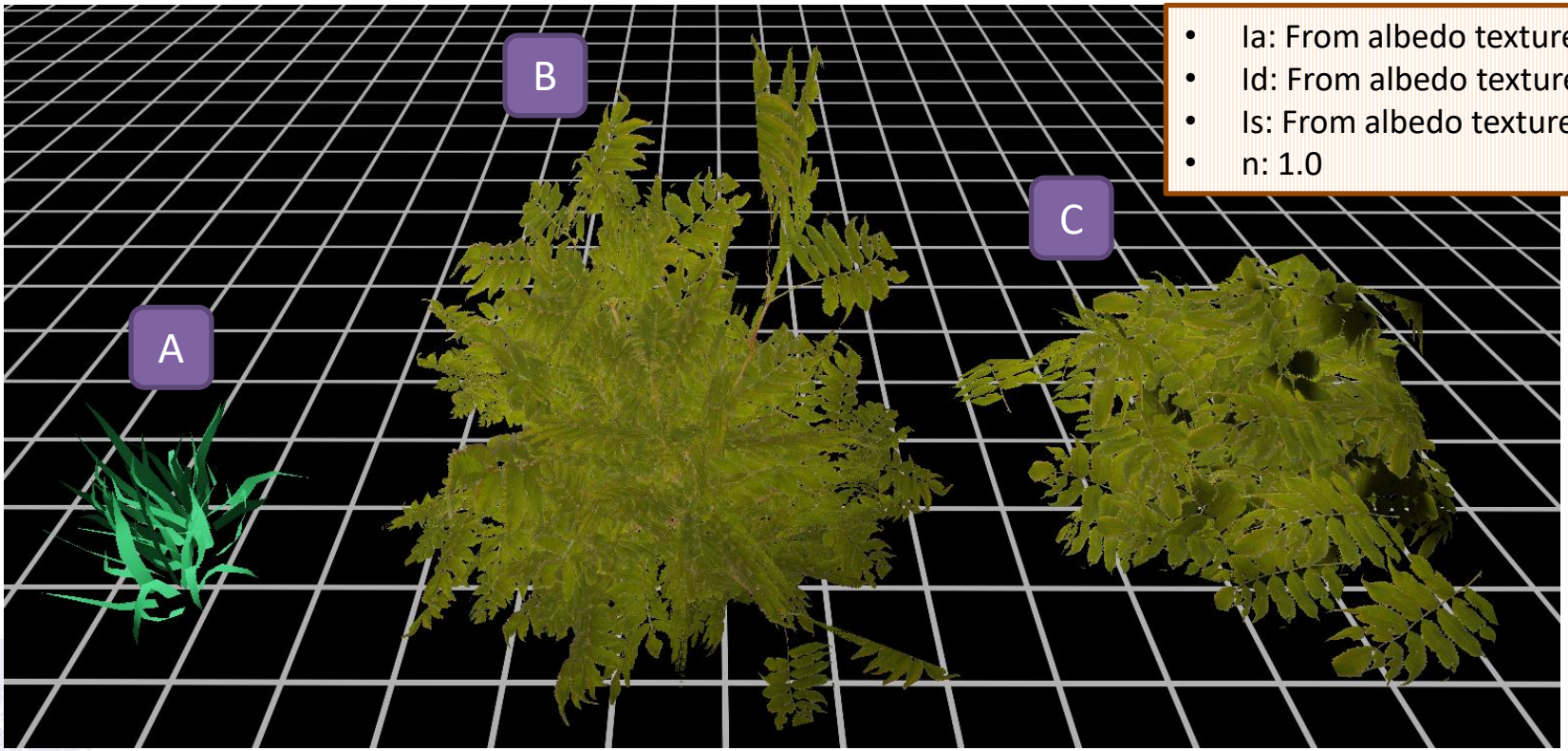
If using blending, we need to sort the instance based on their depth, so here we simply discard the transparent texel



Instance Data

```
#include “src\Scene\SpatialSample.h”

using namespace INANOA::SCENE::EXPERIMENTAL;

void initialize(){
  // initialize the spatial samples
  SpatialSample* sample0 = SpatialSample::importBinaryFile(“poissonPoints_155304s.ss2”);
  SpatialSample* sample1 = SpatialSample::importBinaryFile(“poissonPoints_1010s.ss2”);
  SpatialSample* sample2 = SpatialSample::importBinaryFile(“poissonPoints_2797s.ss2”);

  // get number of sample
  const int NUM_SAMPLE = sample0->numSample();

  // query the position
  for(int idx=0 ; idx<NUM_SAMPLE ; idx++){
    const float* POSITION_BUFFER = sample0->position(idx);
  }
}
```

Plants and Corresponding Data

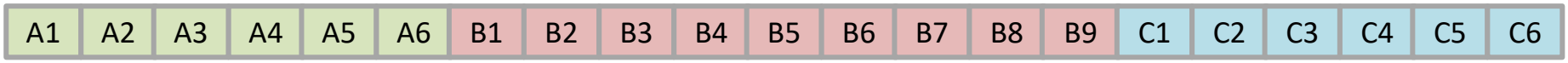


- Ia: From albedo texture
- Id: From albedo texture
- Is: From albedo texture
- n: 1.0

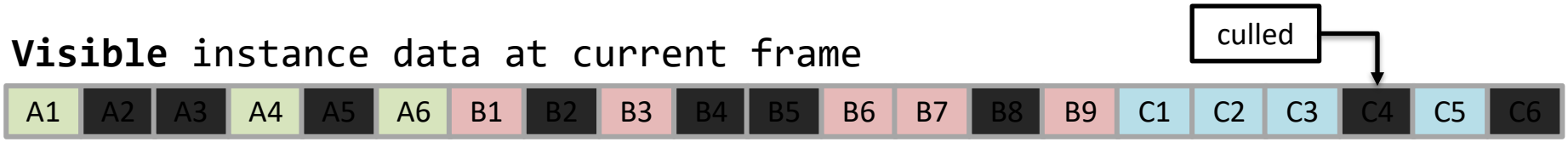
	A	B	C
Mesh	grassB.obj	bush01_lod2.obj	bush05_lod2.obj
Texture	grassB_albedo.png	bush01.png	bush05.png
Poisson sample	poissonPoints_155304s.ss2	poissonPoints_1010s. ss2	poissonPoints_2797s. ss2

Hint: Avoid Chaos Instance Data

Raw instance data



Visible instance data at current frame



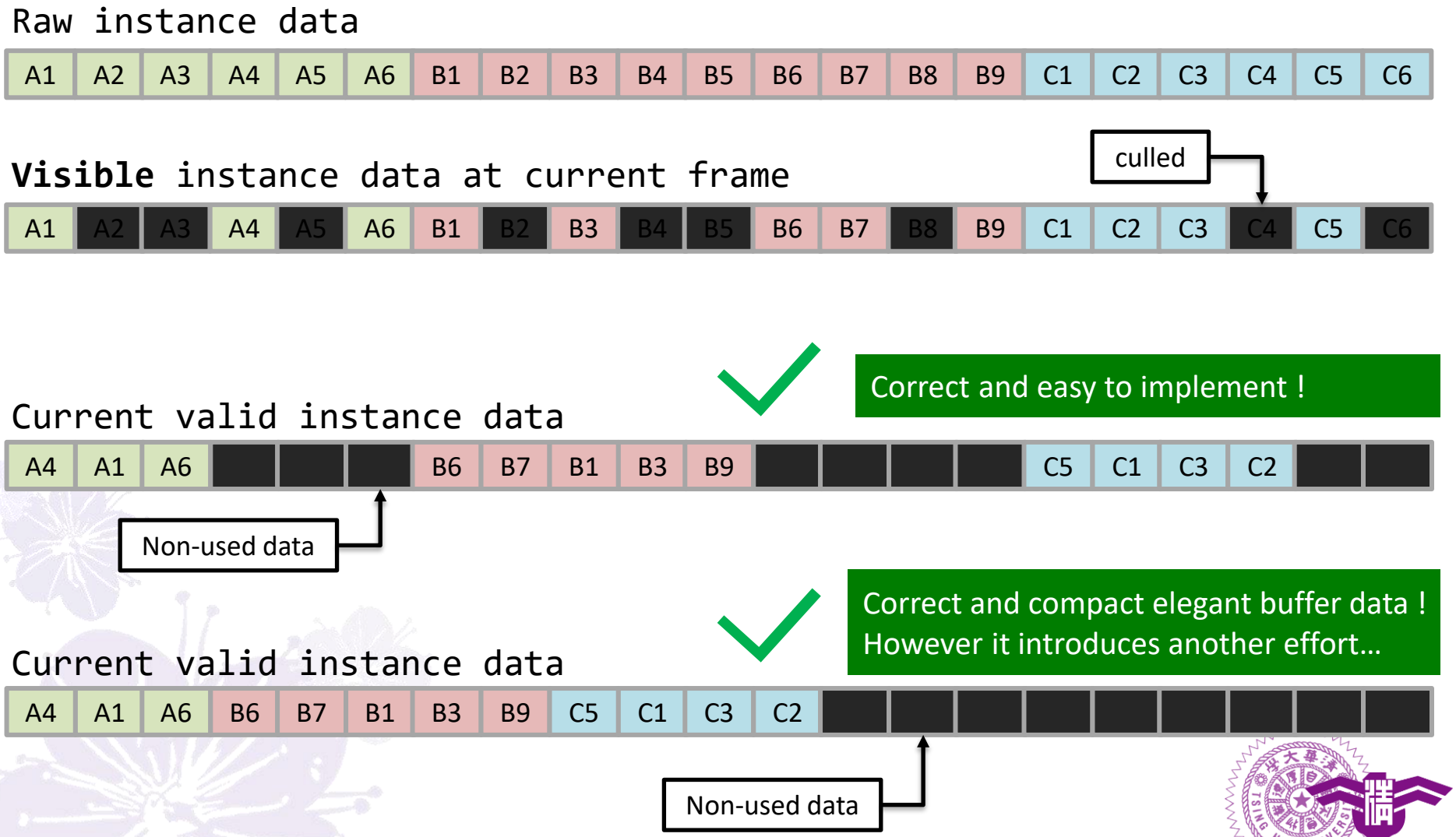
If you simply collect data with this program…

```
void main() {
  const uint idx = gl_GlobalInvocationID.x;
  if(/*visible*/){
    // put data into valid-instance buffer
    // also update the instance count
    const int UNIQUE_IDX = atomicAdd(commands[0].instanceCount, 1)
    currValidInstanceProps[UNIQUE_IDX] = rawInstanceProps[idx].position [idx] ;
  }
}
```

Current valid instance data

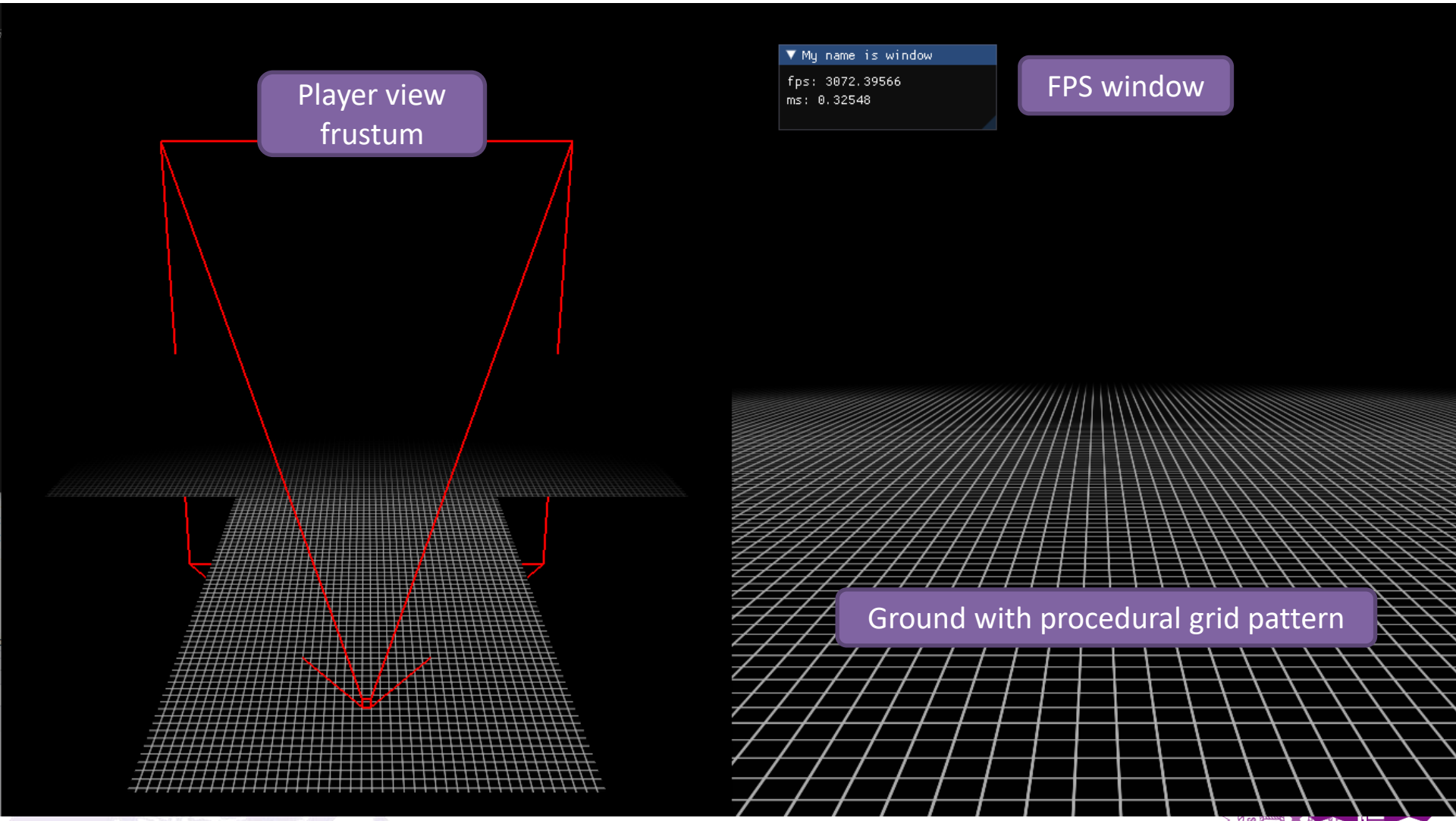
ERROR !!
The instance data of same mesh should be grouped together

Hint: Avoid Chaos Instance Data



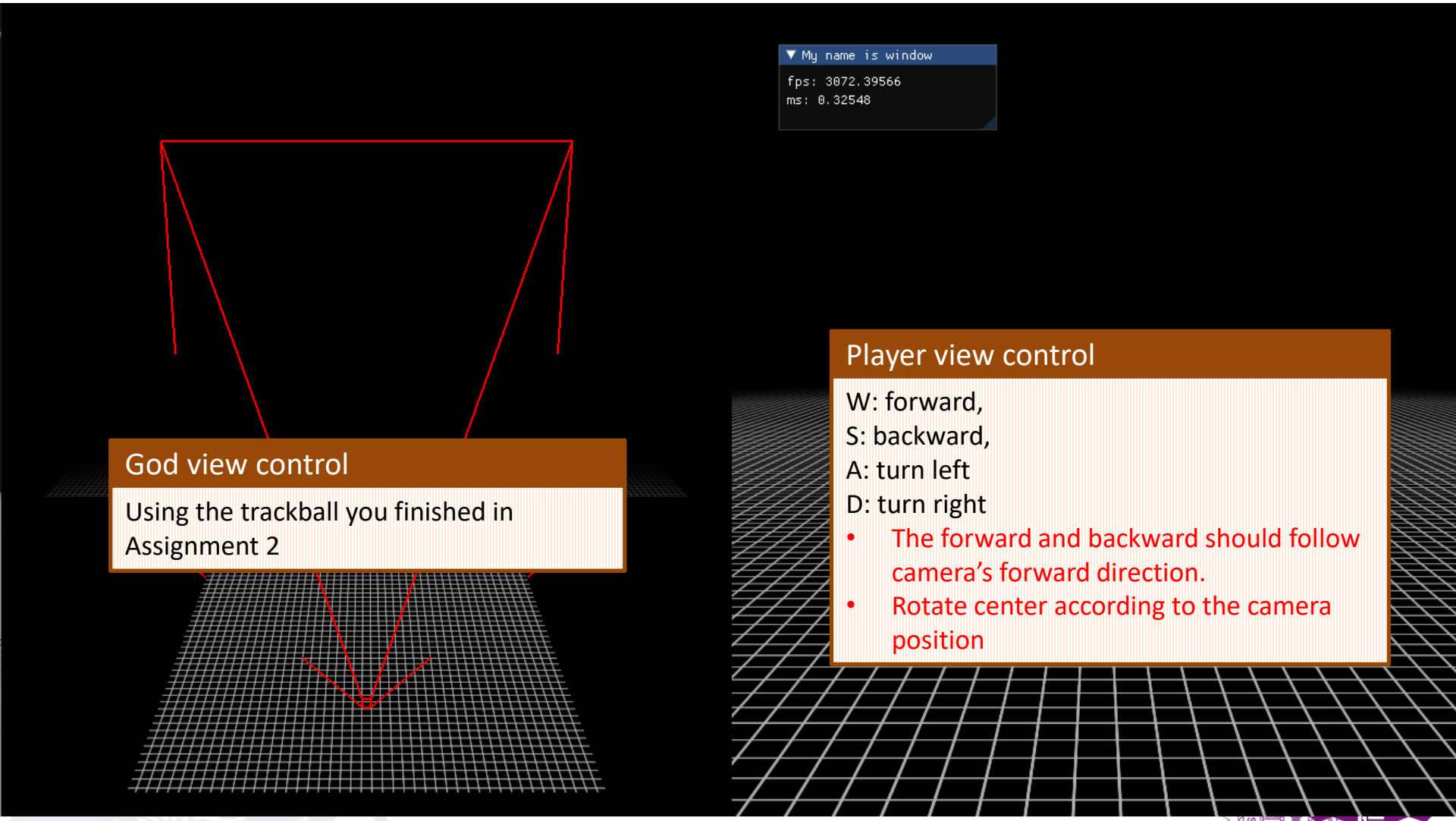
Framework

Platform: x64



Camera Control

Platform: x64



God view control

Using the trackball you finished in Assignment 2

Player view control

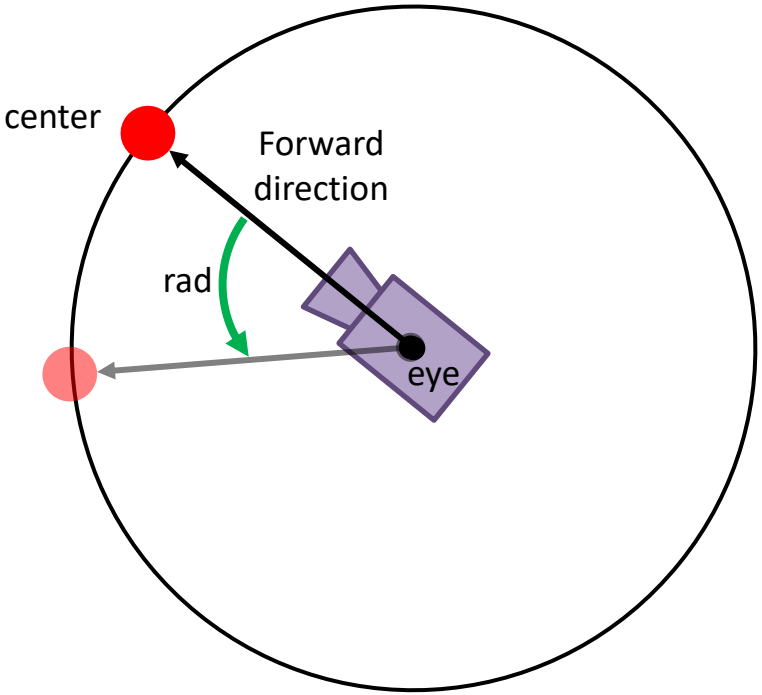
W: forward,
S: backward,
A: turn left
D: turn right

- The forward and backward should follow camera's forward direction.
- Rotate center according to the camera position

god view

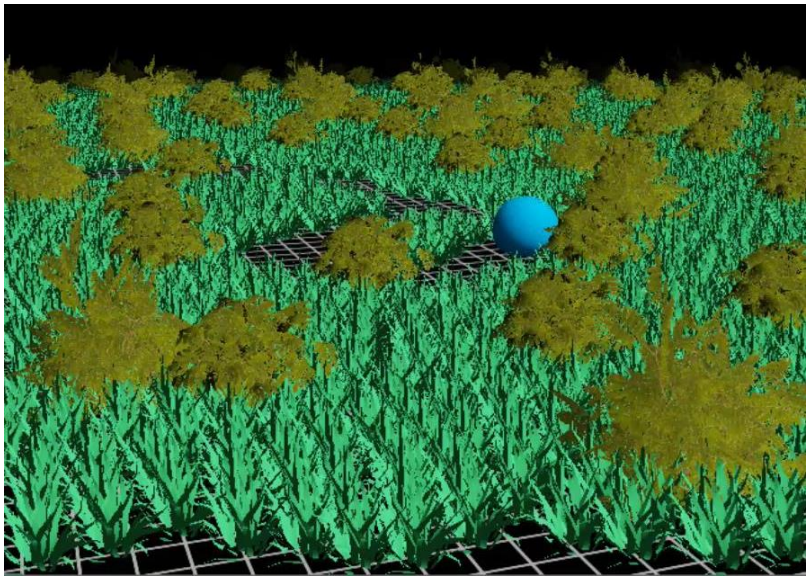
player view

Camera Rotation Control




```
glm::vec3 rotateCenterAccordingToEye(const glm::vec3& center, const glm::vec3& eye,
  const glm::mat4& viewMat, const float rad) {
  glm::mat4 vt = glm::transpose(viewMat);
  glm::vec4 yAxisVec4 = vt[1];
  glm::vec3 yAxis(yAxisVec4.x, yAxisVec4.y, yAxisVec4.z);
  glm::quat q = glm::angleAxis(rad, yAxis);
  glm::mat4 rotMat = glm::toMat4(q);
  glm::vec3 p = center - eye;
  glm::vec4 resP = rotMat * glm::vec4(p.x, p.y, p.z, 1.0);
  return glm::vec3(resP.x + eye.x, resP.y + eye.y, resP.z + eye.z);
}
```

Slime



- Ia: From albedo texture
- Id: From albedo texture
- Is: From albedo texture
- n: 1.0

```
#include “src\scene\Trajectory.h”
INANOA::SCENE::EXPERIMENTAL trajectory ;

void render(){
  // update the trajectory
  trajectory.update() ;
  // get the position of the slime in current frame
  glm::vec3 position = trajectory.position();
  glm::vec4 positionVec4 = trajectory.positionVec4() ;
}
```

Hybrid Render API

- You **can** render procedural plane, view frustum and Slime with **glDrawElements(…)**
- But you **must** render foliage with **glMultiDrawElementsIndirect(…)**

```
void render(){
  // ========== Procedural plane and view frustum ==========
  …

  // ========== Slime part ==========
  // update Slime's position
  …
  // render Slime
  glDrawElements(…) ;

  // ========== Foliage part ==========
  // Use compute shaders to update the instance data buffer
  …
  // render foliage
  …
  glMultiDrawElementsIndirect(…)
}
```

Evaluation

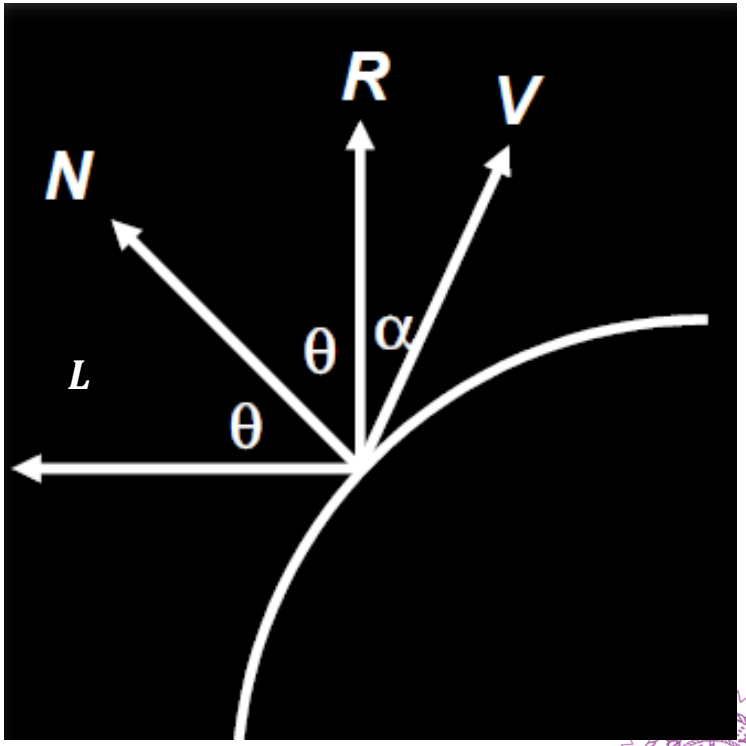
- Vsync disabled
- Resolution: 1344×756

Item	Score
Render the foliage • Masking the transparent part • Phong shading • Culling the instance outside the view frustum with compute shader • FPS > 164 (Running on the 2080super GPU)	70%
Erasing the instance that lies on the Slime's trajectory • Erasing the instance with compute shader • FPS > 164 (Running on the 2080super GPU) Render the slime • Phong shading	20%
Use texture 2D array	5%
Report	5%

GPU	RTX 4070ti	RTX 3090	RTX 2080super	RTX 3070ti (msi laptop)
FPS	660	735	440	280

Phong Shading

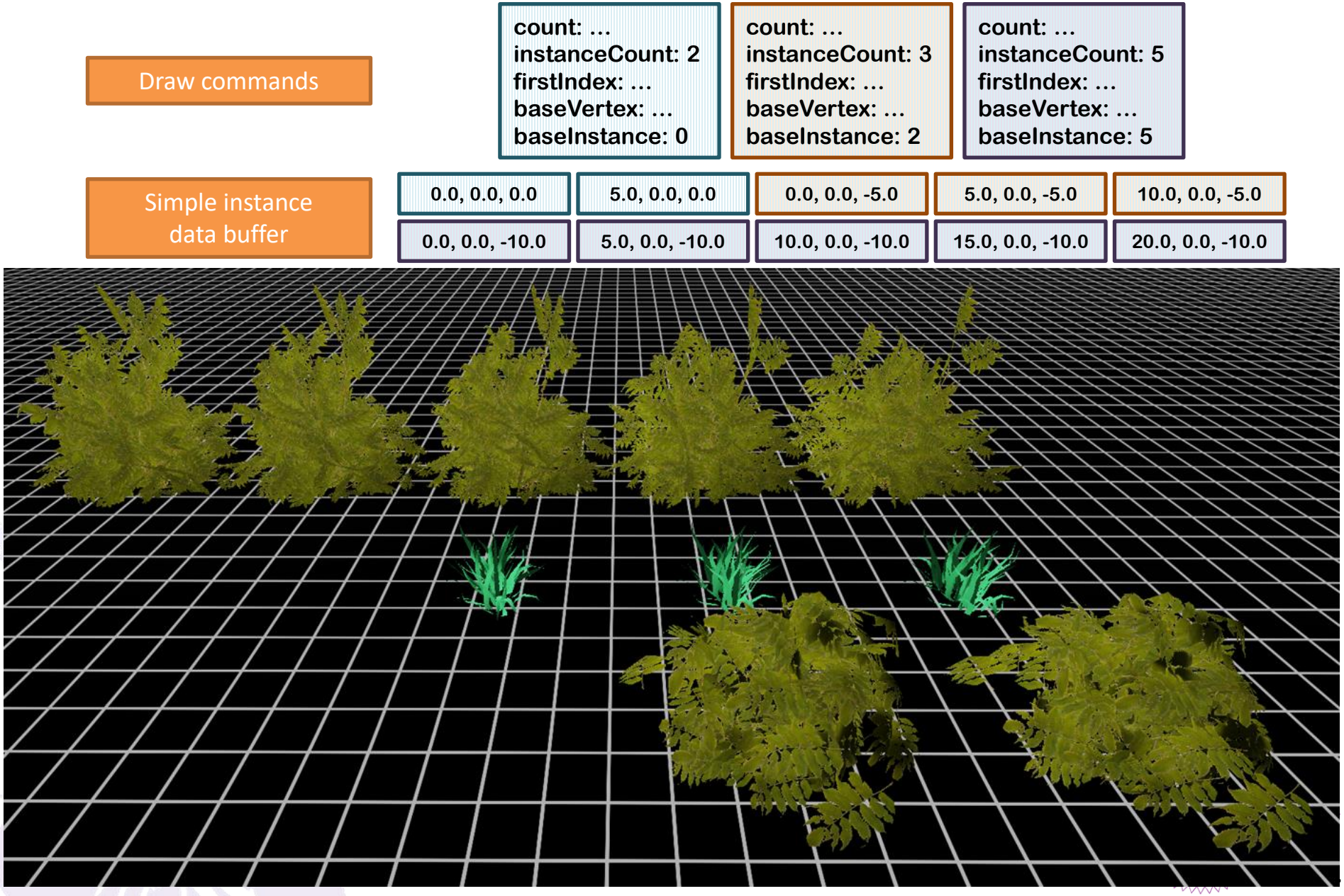
- Directional light direction (L): $(0.3, 0.7, 0.5)$
- Ka: $(0.1, 0.1, 0.1)$
- Kd: $(0.8, 0.8, 0.8)$
- Ks: $(0.1, 0.1, 0.1)$
- Intensity: $(1.0, 1.0, 1.0)$



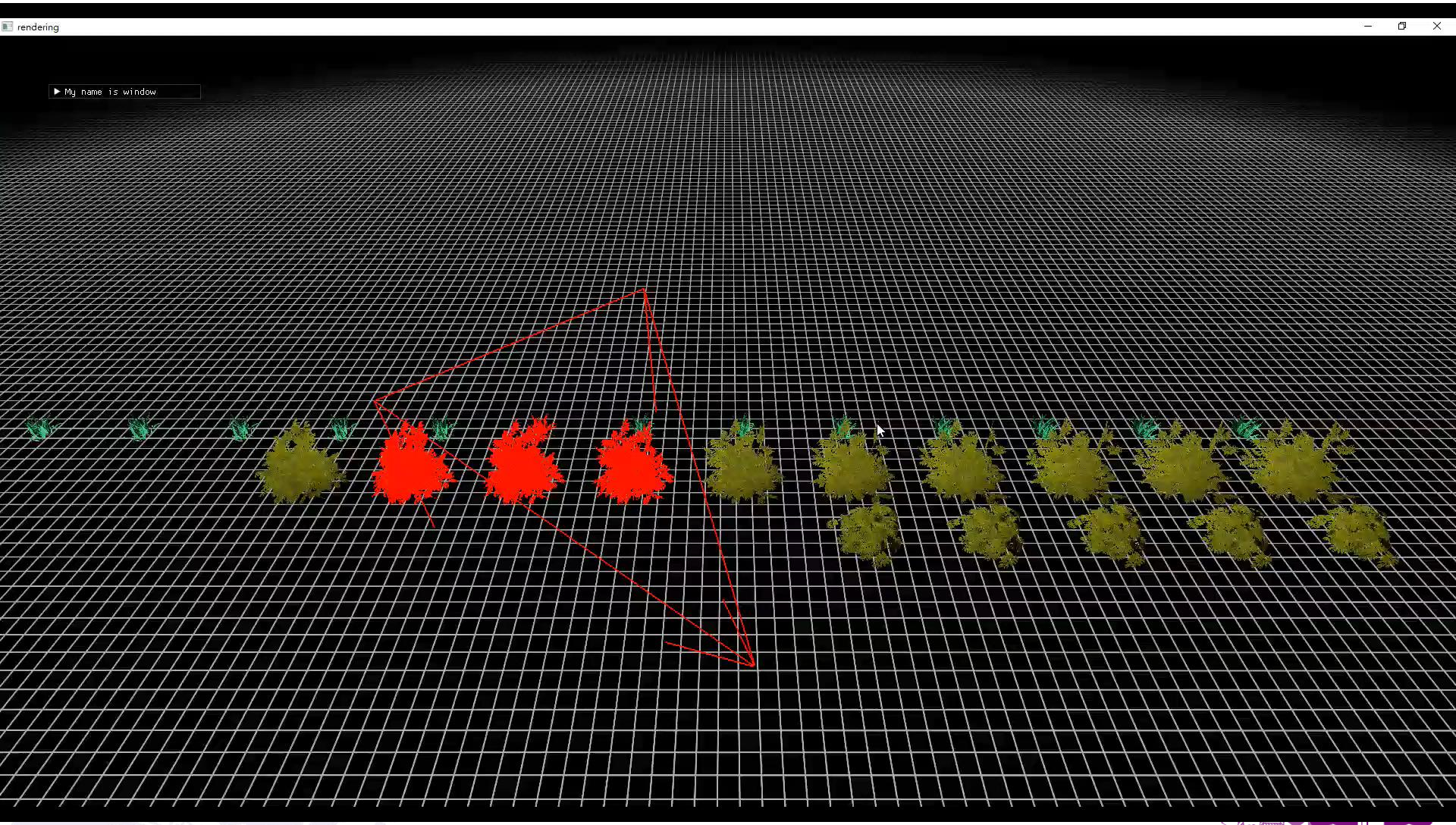
Hint

- The most important first step: **make sure you can draw the foliage successfully with Texture2DArray**
- Make sure the SSBO are set up successfully
  - You can render the scene with **STATIC** SSBO without compute shader
- Per object rendering → Multi object rendering with single draw call
  - glDrawElements → glMultiDrawElements
- Direct rendering → Indirect rendering → GPU driven rendering
  - glMultiDrawElements → glMultiDrawElementsIndirect → Compute shader + glMultiDrawElementsIndirect
- Simple culling rule → Complex culling rule
- Use less and controllable instance data for debugging

- Use less and controllable instance data for debugging



Use marker instead of directly cull the instance when you are debugging



Hint: Step by Step

- Compute shader **absence** indirect rendering
  - Put your instance data to the SSBO (let's call it target-SSBO)
  - Create the **STATIC** draw commands in CPU and then put it into SSBO
  - Bind your draw commands SSBO to GL_DRAW_INDIRECT_BUFFER target
  - Render the scene with **glMultiDrawElementsIndirect(…)**
  - Change your draw command parameters **in client side (CPU)** to confirm whether it works
    E.g. instanceCount
- Compute shader **presence** indirect rendering
  - Prepare the SSBO for storing **raw-instance-data** (let's call it source-SSBO)
  - **Simply copying the instance data from source-SSBO to target-SSBO in your compute shader,** if every work well, the rendered image would be diff
  - Apply the culling rule in your compute shader to dynamically update the target-SSBO

Assignment 3

- Announce date: 2024/10/28
- **Due date: 2024/11/18 23:59**
- Compress your project and executable and report as ZIP and Upload to FTP
- Submit MD5 checksum
- **Failure to comply with our rules will result in a 10-point deduction.**
- **Please remove unused code in the framework!**

Assignment 3

- **You should upload your (probably large) final product to FTP**
- FTP (no downloading, no deletion)
- Server: cgv.cs.nthu.edu.tw
- Account: GPA2024
- Password: 2024GPA
- Folder: Assignment3
- Use your student id to create a folder and put your files in it (to modify, create a new one with _v3, for example: 104062517_AS3_v3.zip)
- Everyone have to submit MD5 checksum

MD5 checksum

- 除了作業本身外，請透過 MD5 獲得作業checksum 後填入 google 表單
- 如遇各種原因無法在作業期限前完成上傳的同學，我們將比對 checksum。若 checksum 一致則不算遲交。
- 若有多個 checksum 則取時間最晚的為主。
- MD5 online generator
- MD5 checksum 登記表單
- MD5 checksum 登記查看
- MD5 使用方法

Assignment 3

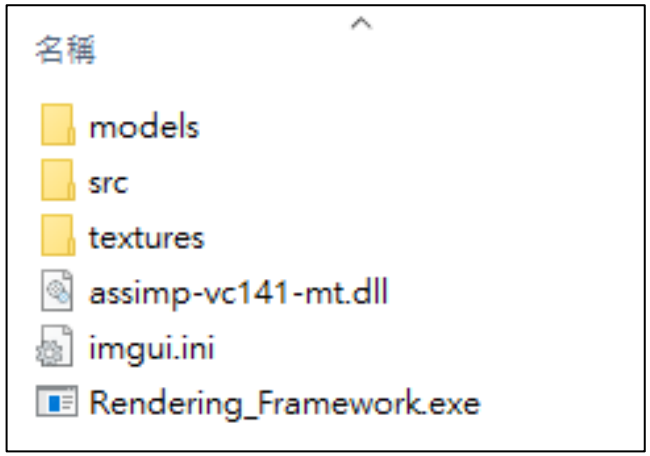
- You will get **0** point when you…
  - **DON'T** use compute shader to cull your foliage instances
  - **DON'T** use glMultiDrawElementsIndirect to render your foliage instances
  - **DON'T** allow TAs to control your god camera and player camera
  - Your program **CANNOT** be resized

抄襲規則

- 抄襲者與被抄襲者皆以 0 分計算
- 抄襲者定義
  - 採用現有資源(線上下載、同學(or 學長姐)的code)且不經修改直接繳交作業者
- 不接受事後補交
- 以程式碼相似度比對工具比對 (Stanford Moss)，若相似度達 20% 視為抄襲

Assignment 3

- **Find the compiled .exe in x64/Release folder**
  - **Make sure your program can find the shader codes and all necessary assets (.obj, .png …)**
- **Make sure your .exe can run by simply clicking it!**



Report Format

- Name your file 學號_AS3_Report.pdf
- Required content:
  - screenshot of your window with character/ scene in it, with different post effect
  - Functions in your program/how to use, which IDE and its version do you use, etc.
  - Only 5%, writing a lot won't get you more!