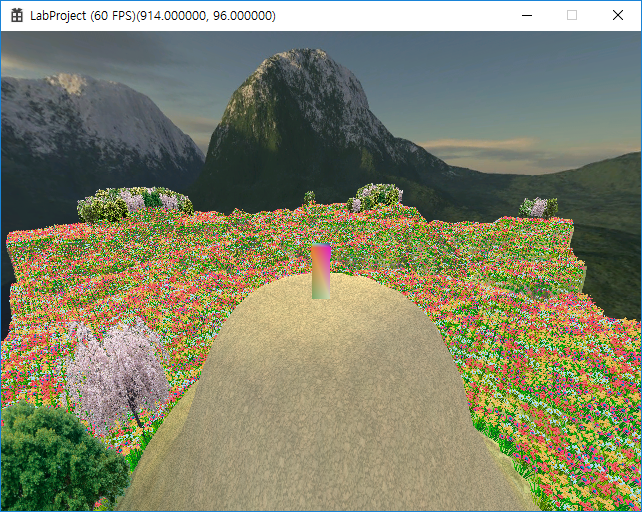
**3D 게임프로그래밍2**

**과제 1**

**게임공학과 2013180013 박진수**



[실행 화면]

**해당 과제는 LabProject08-7을 기반으로 작성되었으며, 조작법도 동일하다.**

**과제 목표**

꽃, 나무 오브젝트를 **빌보드 오브젝트**로 생성.

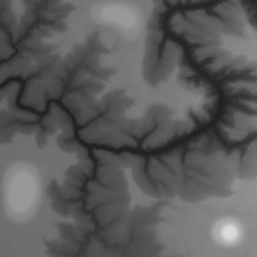
**나무 텍스처의 종류**는 **5가지 이상.**

어떤 지점은 나무가 많이 배치 되어있으며, 어떤 지점은 나무가 없어야 한다.

**평지에는 꽃을 배치할 수 있는 만큼 배치**하며 꽃은 바람?에 살랑살랑 흔들린다.

**지형은 1000x1000 이상**이 되어야 하며, **흙밭과 돌밭을 구분**하도록 한다.

**사용한 리소스**

**Height Map Batch Map**

    
**Base Texture Detail Texture 1 Detail Texture 2**



**Flower Textures**



**Tree Textures**

**프로그램 구조**

Device, Command Queue, Command List,

Render Target View, Depth Stencil View, Swap Chain 등 생성

Camera 소멸

Mesh, Shader 소멸

Player 소멸

Scene의 모든 객체 소멸

Scene 소멸

게임 월드 소멸

(Release Objects)

프레임 단위로 게임 월드 갱신

(Animate, Rendering, Process Input)

게임 프레임워크 소멸

(On Destroy)

프로그램 종료

(Exit)

Scene의 모든 객체들을 렌더링

Player를 렌더링

Scene의 모든 객체들을 갱신

키보드, 마우스 입력에 따라

Player, Camera 갱신

Player Update

Animate Objects

Objects Rendering

Scene 생성

Player 생성

게임 월드 생성

(Build Objects)

게임 프레임워크 생성

(On Create)

프로그램 실행

(Execute)

Root Signature 생성

Height Map Terrain 생성

SkyBox 생성

Flower Objects Shader 생성

Tree Objects Shader 생성

Camera 생성

Player Mesh 생성

Player Shader 생성

Root Signature 릴리즈

Height Map Terrain 소멸

SkyBox 소멸

Flower Objects Shader 소멸

Tree Objects Shader 소멸

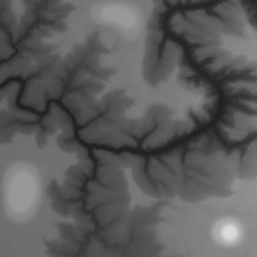
Device, Command Queue, Command List,

Render Target View, Depth Stencil View, Swap Chain 등 소멸

**프로그램 구조 – 루트 시그너처**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Root Parameter** |  | **Descriptor Range** | | |
| **0** | **CBV**  [b0]  (플레이어 월드 변환 행렬) | **-** | **-** | | |
| **1** | **CBV**  [b1]  (카메라 변환 행렬)  (투영 변환 행렬) | **-** | **-** | | |
| **2** | **DESCRIPTOR TABLE** | **0** | **CBV**  [b2]  오브젝트 월드 변환 행렬 | | |
| **3** | **SRV**  [t0]  구조화된 버퍼  오브젝트들의 정보  (월드 변환 행렬, 텍스처 종류) | **-** | **-** | | |
| **4** | **DESCRIPTOR TABLE** | **1** | **SRV**  [ t1 ~ t3 ]  지형 텍스처의 배열 | t1 | Base Texture |
| t2 | Detail Texture 1 |
| t3 | Detail Texture 2 |
| **5** | **DESCRIPTOR TABLE** | **2** | **SRV**  [t4]  오브젝트 텍스처 | | |
| **6** | **DESCRIPTOR TABLE** | **3** | **SRV**  [ t5 ~ t9 ]  텍스처의 배열 | t5 | Texture 1 |
| t6 | Texture 2 |
| t7 | Texture 3 |
| t8 | Texture 4 |
| t9 | Texture 5 |

**프로그램 구조 – Height Map Terrain 생성**



**Height Map Batch Map**

**257x257 크기의 높이맵 이미지를 4배 곱하여 1028x1028 크기의 지형을 생성하였다.**

**지형을 생성하면서 오브젝트(나무, 꽃)들을 배치하기 위한 이미지를 따로 만들어 높이맵을 읽어들이는 방식과 같은 방식으로 이미지를 읽어들여 각 픽셀에 대한 정보를 저장하였다.**

**해당 픽셀값이 n < 200이면 꽃 오브젝트가 배치되도록 하였고 100 < n < 200이면 나무 오브젝트가 배치되도록 하였다.**

**따라서 꽃 오브젝트는 (흰색)이 아닌 부분에, 나무 오브젝트는 (회색)인 부분에 모두 배치되도록 했고, (흰색)인 부분에는 아무런 오브젝트가 배치되지 않게 되도록 했다.**

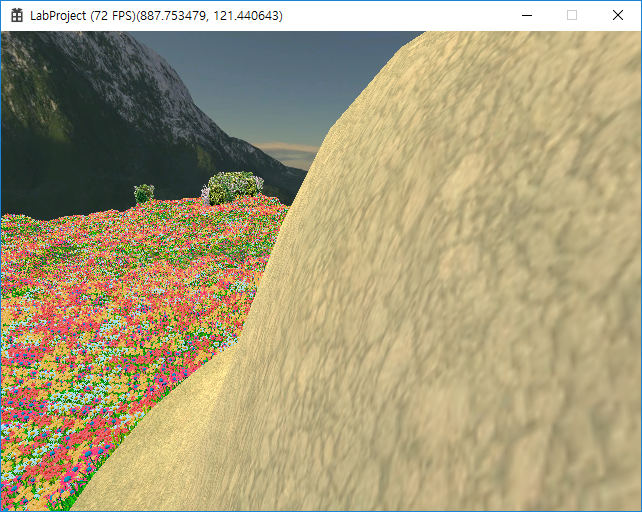
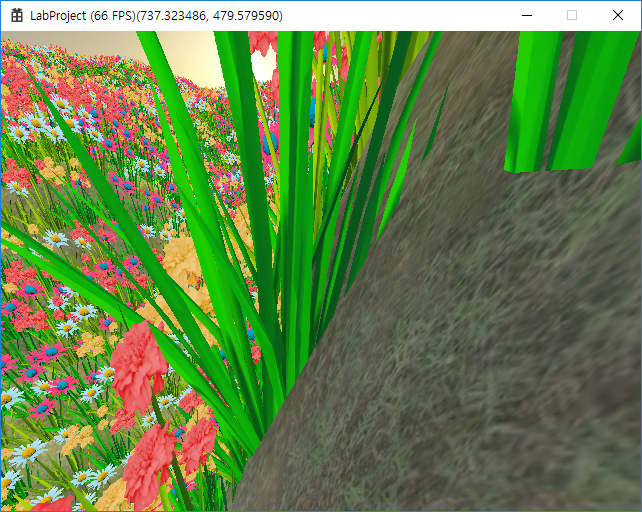


**Base Texture Detail Texture 1 Detail Texture 2**

**Base Texture는 지형 전체 크기로 매핑, Detail Texture는 지형 타일마다 매핑된다.**

**단, Detail Texture는 배치맵을 이용해서 해당 정점 위치의 픽셀값이 n > 200인 부분에는 Detail Texture2가 매핑 되도록 해서 돌밭을 표현했고 나머지 픽셀값에는 Detail Texture1이 매핑 되도록 해서 흙밭을 표현했다.**

**각 정점 정보로는 위치, 색상, Base Texture UV, Detail Texture UV, Detail Texture Number를 저장했고 텍스처의 배열을 이용해 Detail Texture Number를 인덱스로 활용하는 형태로 하여 Detail Texture를 적용시켰다.**



**[Detail Texture 1이 적용된 화면] [Detail Texture 2가 적용된 화면]**

**프로그램 구조 – 오브젝트 생성 ( Flower Object Shader, Tree Object Shader )**



**꽃 오브젝트를 생성할 때 지형의 크기를 고려해서 가로 250, 세로 250, 총 62,500개의 오브젝트를 생성하되 배치맵을 이용해서 62,500 – a ( 픽셀값이 n <= 200인 부분 )의 수만큼 배치하도록 했다.**



**나무 오브젝트도 같은 방식으로 가로 50, 세로 50, 총 2,500개의 오브젝트를 생성하되 2,500 - a ( 픽셀값이 100 < n < 200 인 부분 )의 수만큼 배치하도록 했다.**

**빌보드 오브젝트에 텍스처를 적용하기 위해 텍스처의 배열을 사용했다. 그러기 위해서 각 오브젝트를 생성할 때 텍스처의 배열의 인덱스로 사용할 값을 rand()%n으로 정해 쉐이더에 넘겨주었다.**

**프로그램 구조 – 오브젝트 갱신 ( Animate Objects )**

**빌보드 오브젝트는 항상 카메라를 향해서 Look 벡터를 가져야 하기 때문에 오브젝트를 갱신할 때마다 Up 벡터는 (0.0, 1.0, 0.0)을 고정으로 하며 Look 벡터만 카메라의 위치와 오브젝트의 위치를 벡터의 뺄셈으로 구해 정규화하여 Look벡터를 적용, 그 후 Up 벡터와 Look 벡터를 외적하여 Right 벡터를 구해 오브젝트가 항상 카메라를 바라보도록 했다.**

**꽃 오브젝트가 바람에 살랑살랑거리는 모습을 표현하기 위해 꽃 오브젝트를 생성할 때 먼저 기본 회전 각도를 0 ~ 1.5 사이로 정해주고 매 프레임마다 회전 각도에 더하거나 빼는 변수를 정해준 후 회전 각도에 적용하여 -1.5 ~ 1.5 사이의 값만 갖도록 했다.**

**프로그램 구조 – 렌더링 ( Render )**

**지형 ( Terrain )**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **HLSL** | |  | **서술자 힙** |  | **루트 시그너처** |
| **ConstantBuffer** | **GameObjectInfo**  **[b2]**  WorldTranf |  | **CBV** |  | **…** |
| **Texture2D** | **TextureArray[0]**  **[t1]**  BaseTexture |  | **SRV** |  | **서술자 핸들** |
| **TextureArray[1]**  **[t2]**  DetailTexture1 |  | **SRV** |  | **…** |
| **TextureArray[2]**  **[t3]**  DetailTexture2 |  | **SRV** |  | **…** |

**지형을 렌더링하기 위해 지형을 생성할 때 위치 정보를 프레임마다 넘겨주기 위해 업로드 힙을 만들고 매핑을 하여 상수 버퍼 뷰와 연결 후 상수 버퍼 뷰를 서술자 힙에 연결하였고 텍스처 정보도 CreateTextureResourceFromFile() 함수를 통해 만들어 쉐이더 리소스 뷰와 연결, 그 후 서술자 힙에 연결했다. 그리고 입력 조립기에 정점 위치, 색상, Base Texture UV, Detail Texture UV, Detail Texture Number를 보내 HLSL에서 이것들을 사용해 렌더링하도록 했다.**

**빌보드 오브젝트 ( Billboard Objects )**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **HLSL** | |  | | **서술자 힙** |  | **루트 시그너처** |
| **Texture2D** | **TextureArray[0]**  **[t5]**  Texture1 |  | | **SRV** |  | **…** |
| **TextureArray[1]**  **[t6]**  Texture2 |  | | **SRV** |  | **서술자 핸들** |
| **TextureArray[2]**  **[t7]**  Texture3 |  | | **SRV** |  | **…** |
| **TextureArray[3]**  **[t8]**  Texture4 |  | | **SRV** |  |  |
| **TextureArray[4]**  **[t9]**  Texture5 |  | | **SRV** |  |  |
|  | |  | |  |  |  |
| **StructuredBuffer** | **GameObjectsInfo**  **[t0]**  WorldTransf  nTexture | |  |  |  | **SRV** |

**빌보드 오브젝트들을 렌더링하기 위해 인스턴싱을 사용하였다.**

**위치 정보와 텍스처 배열 인덱스 정보를 쉐이더 클래스에서 일괄적으로 갱신시키기 위해 오브젝트 개수와 보내줄 정보를 가진 구조체를 곱한 길이를 가진 업로드 힙을 만들고 매핑하였다.**

**그렇게 업로드 힙을 만들어 나중에 렌더링 시 모든 오브젝트의 위치, 배열 인덱스 정보를 업로드 힙에 갱신한 후 쉐이더 리소스 뷰(구조화된 버퍼)에 연결하였다. 구조화된 버퍼에 모든 오브젝트에 대한 정보가 담겨있으므로, 한 번만 Draw\*() 함수를 호출하여 프레임 레이트가 잘 나오면서도 많은 객체를 그릴 수 있도록 했다.**

**텍스처 정보도 CreateTextureResourceFromFile() 함수를 통해 만들어 쉐이더 리소스 뷰와 연결, 그 후 서술자 힙에 연결했다. 그리고 입력 조립기에 정점 위치와 Texture UV만 보내 HLSL에서 이것들을 사용해 렌더링하도록 했다.**