In this assignment, I researched how to simulate the wave of the water. In the beginning, I wanna use C# to control the changes of every pixel. However, it is not convenient and efficient. So, I learned how to use a new weapon “Shader Graph” to design water in Unity 3D. In this process, I read the document online. Even though it is really interesting and simple, it can’t satisfy my requirement, because I will make many calculate to get the real-time change of the UV. So, finally, I found some books to learn CG language and how GPU works.

In order to understand my programs, you should learn some basic knowledge about the process in rending[1]. This part is also important for my following assignments. The procedure in rending has a strict rule. There are 4 kinds of default shaders in Unity. According to my requirements in this week, I chosen Unlit shader to simulate water (Maybe I will change it in the next assignment because water is transparent).

[1] https://unity3d.com/cn/learn/tutorials/topics/graphics/gentle-introduction-shaders

So, let’s talk about how to simulate the wave of the water in Unity 3D. I designed 2 different ways of waves’ moving. They are all presented as the shader for the material on a plane(water surface).

Thus, the first thing we need to do is to create the Unlit Shader in a folder in Assets (Right Click--Create--Shader--Unlit Shader). When you double click it, you can open it in Visual Studio. You will found it has been programmed. Now, according to our requirements, we can add our own programs in rending procedure.

1. For the flowing water (used in the river), the first thing I did is to let the UV change according to the time.

```
float2 flowSpeed = float2(_SpeedX * _Time.y, _SpeedY * _Time.y);

// “flowSpeed” is the vector that means the flowing direction and flowing speed of the water.
// “_Time” is a default float4 various, (t/20, t, t*2, t*3). It is used to animate things inside the shaders.
/ /“_SpeedX” and “_SpeedY” are the various default by my self means the speed and the direction of water on x-axis and y-axis.
```

Now, when you the load the shader in a new material and put the material on a plane, you will find it is rigid, likes sliding advertisements. In order to make it better, we need to add some noise for the water. Thus, the second thing we need to do is to set the time-varying offsets by using noise picture (gray scale picture). In my program, the direction of the offset is (1, 1).

```
float noise = tex2D(_NoiseTex, uv).r * _Intensity;
```

```
float2 speed = normalize(float2(1.0,1.0));
float4 col = tex2D(_MainTex, uv + flowSpeed + noise * float2(sin(_Time.y * speed.x),
sin(_Time.y * speed.y))*_WaveScale);


// "tex2D" is a default function that get the value of pixel according to the coordinate
of UV.
// "_NoiseTex" is the sampler of noise (gray scale picture).
// "_Intensity" and "_WaveScale " means the degree how noise picture influence the
wave.
// "_MainTex" is the sampler of water (UV picture).
```

What is worth mentioning is that I used sine function to simulate the noise changes, because it is period, continuous, and works in [-1,1].

Then, you can load your shader in a new material and create a plane in Hierarchy. Drag your material on this plane, you can get the river water.

2. For clam water, we should consider 2 things, overall fluctuation and waves from objects.

```
float noise = tex2D(_NoiseTex, uv).r * _Intensity;
float2 uvDir = normalize(uv - fixed2(0.5, 0.7));
float dis = distance(uv, fixed2(0.5, 0.7));
float4 col = tex2D(_MainTex, uv + noise * float2(sin(_Time.y * speed.x), sin(_Time.y
* speed.y)) + _WaveStrength * uvDir*sin(_Time.y*_TimeScale + dis *
_WaveFactor)/5/dis);
```

Overall fluctuation is similar with the noise in the flowing river we talked about just now.

For the waves from objects. I set the position of the object is (0.5,0.7), and then calculated the related data of different pixel, including direction and distance. Then I used sine function to simulate the ripples. The ripples will decrease according to the distance's increasing.