

BScCM Final Year Project 2021 – 2022

Phase II Final Report

<< Group Name: KANGMISAMA; Group No.: 16 >>

<< Azalea >>

Student Name/EID : XIAO Bushi(Group Leader) / bushixiao2

55670545

Programme Code : SM4701/4712B

Supervisors	S C : Dr. WEI, Shiyu Louisa M _____ C : S : Dr. Wang Shiqi _____
Date	: Day April 29 _____

1 Introduction	3
1.1 Objective	3
1.2 Deliverables	4
2 Work Plan/Milestones/Gantt chart	4
3 Background research	6
4 Framework	8
4.1 concept	8
4.2 Storyboard	10
5 Implementation and evaluation	11
5.1 Rigging	11
5.2 Material	14
5.3 Shader	20
5.4 Other 3D scene models	30
5.5 Cloth Simulation	33
6 Conclusion: critical evaluation of the project	36
6.1 limitations and future developments	37
6.2 Summary	37
6.3 Other Links	38
7 Reference List	39

1 Introduction

This project will be presented in the form of 3D character animation. The length of the video is 4 minutes and ten seconds.

The art style of the animation is similar to the music video of 1925(T-pocket, 2017). It combines the Eastern and Western styles to express that this is an era of the impact and integration of Eastern culture and Western culture. In addition to a combination of art and technology, this project will incorporate other interdisciplinary knowledge, including history, geography, politics, and semiotics.

The theme of the story is anti-war and anti-oppression. The main background of the animation takes place in a virtual East Asian country in the twentieth century (a fusion of all East Asian cultures). The events that occurred are based on real histories, such as the Sino-French War, Showa Revolution, and the Gwangju Democratic Movement... But they did not follow the sequence of time and space in real history, so the endings of the story are different from the real world.

The story is told from the perspective of the heroine Azalea. The lens moves along with Azalea's steps. It begins with Azalea bidding farewell to her father who left home to participate in the revolution. Then, foreign enemies invaded and the feudal government was overthrown until the military government collapsed and the democratically elected government was in power. Symbols, images, and scenes are used to symbolize specific things. The visual expression/storytelling technique of I, Pet Goat II (Heliofant, 2013) can be an appropriate reference.

1.1 Objective

The artistic objective is to use various techniques related to visual art to symbolize a certain era or historical event, such as the Taisho style kimono to represent the ten years of the Taisho

democratic period. There are many metaphors in the content of the animation that reflect realistic historical events, but the sequence of these events is disrupted, leading to different endings. And many symbolic figures will be inserted into the animation based on the study of semiotics. The art style of animation will combine the culture of the three East Asian countries in the early twentieth century.

The technical objective is to use CG technology to establish a complete character model and scene model, complete the binding and animation of the character model, as well as the production of animation 2D and 3D special effects. 3D to 2D rendering technology in making animations, which is also called cartoon/toon rendering. The scene models/character models in the animation are all 3D, but use a special toon shader to render them to look like 2D animation.

1.2 Deliverables

After research and continuous improvement of the script and storyboard, I gradually discovered that there may not be enough time to make it into a game or interactive animation, so the final project will be presented in the form of an animated short film. The submitted format will be H264, which is mp4 video.

2 Work Plan/Milestones/Gantt chart

Production Schedule (Including Pre-Production, Production and Post-Production)													
MM/YY – MM/YY	W k 1	W k 2	W k 3	W k 4	W k 5	W k 6	W k 7	W k 8	W k 9	W k 10	W k 11	W k 12	W k 13

The completed part during the Christmas holiday: character modeling, including bones, rigging, skinning, weights.

The part completed by Semester B: adding characters to the scene, adjusting rendering parameters such as lighting, writing rendering scripts, and making and rendering animation frames.

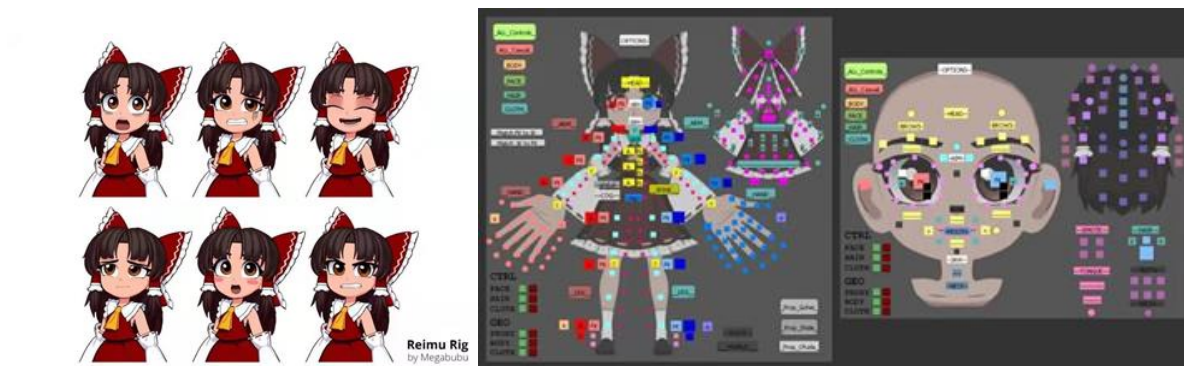
The unfinished part: **it has been rendered once, and there is already a complete animation video**, but this is a 540p rendering, the purpose is to see if there are any bugs in the plot or in the animation. The next job is to modify the final motion, lighting, background sound effects, and special effects, etc., and then do the final rendering.

3 Background research

As early as 2003, Anjyo and Hiramitsu researched the use of shaders to make highlight shapes and animations in cartoon style. This is done by using Blinn's traditional specular reflection model to make the initial highlight shape. Then, it interactively modifies the initial shape through geometry, style, and Boolean transformations until the final desired shape is output. This does not require the use of texture maps, and the current cartoon rendering has been simplified. With the advancement of algorithms and software updates, rendering time is shorter, and users need to configure fewer parameters by themselves, and it is more convenient. Not using texture maps greatly speeds up the efficiency of making models and also speeds up rendering.

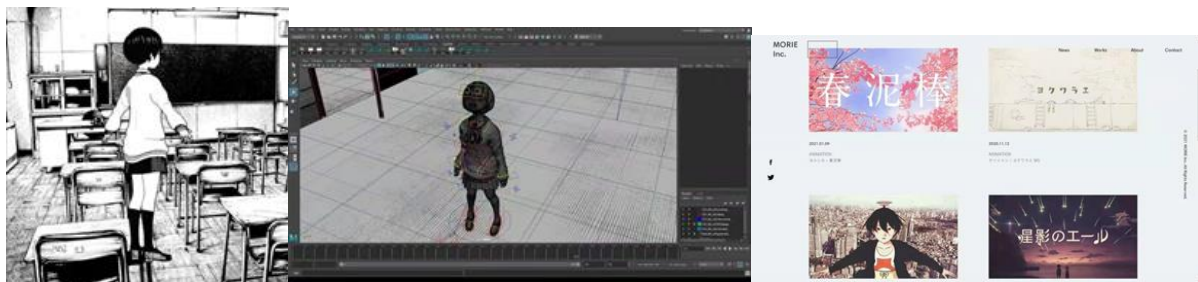
I am more interested in the creation of characters in 3D animation than in the animation itself, and a very important part of it is character rigging. In the 3D animation industry, rendering 3D into 2D effects is also a new aesthetic. Because the 2D characters rendered with 3D technology are more unique, compared to rendering other real 3D characters, this saves rendering time and compared to 2D animation, this saves the time-consuming drawing frame by frame. I have been following the work of the animator Chonlawat Thammawan because he provides some free animation models that are bound together. As a technical animator, he recently released a set of

characters made in Maya with a very simple animation style, such as Reimu's rigging model as shown below.



Rigging of Reimu by Chonlawat Thammawan

With the existing cartoon materials, Maya can easily render various animation-style animations. According to him, this binding is only for offline animation work, because the character is too big to be used in the game. If you reach an ideal state, you can get a more animated character through step tangents. This character not only has a mixed FK/IK (forward dynamics/inverse dynamics) dynamics of the spine and limbs, but the face, hair, and clothes can also be animated. The production of the face and body can be achieved through the AnimSchoolPicker plug-in. Although it can be rendered and output by Maya's hardware renderer, this character is designed to be animated by Maya's Viewport 2.0 and includes the ShaderFX cartoon material with outline width properties.



Some 3D to 2D render productions by company Morie Inc.

Actually 3D rendering 2D animation technology has also been widely used in the industry. For example, the mv produced by Morie. inc production company for many Japanese music

producers is short films of a similar style. Compared with the original 3D animation, they are more distinctive and look more like Japanese manga.

4 Framework

4.1 concept

To create a story of a girl living in a turbulent era, the story is told from the perspective of the heroine Azalea. The lens moves along with Azalea's steps. It begins with Azalea bids farewell to her father who left home to participate in the revolution. Then, foreign enemies invaded and the feudal government was overthrown until the military government collapsed and the democratically elected government was in power. Symbols, images, and scenes are used to symbolize specific things.

Time / Period	Location / Country	Culture / Genre
Since this is a fictitious history, it is not in Japan. The Taisho era is just a reference for style here.	It is also not a specific country, but a visual reference as East Asia style.	War Genre and Historical Drama

Visual Theme & Visualization

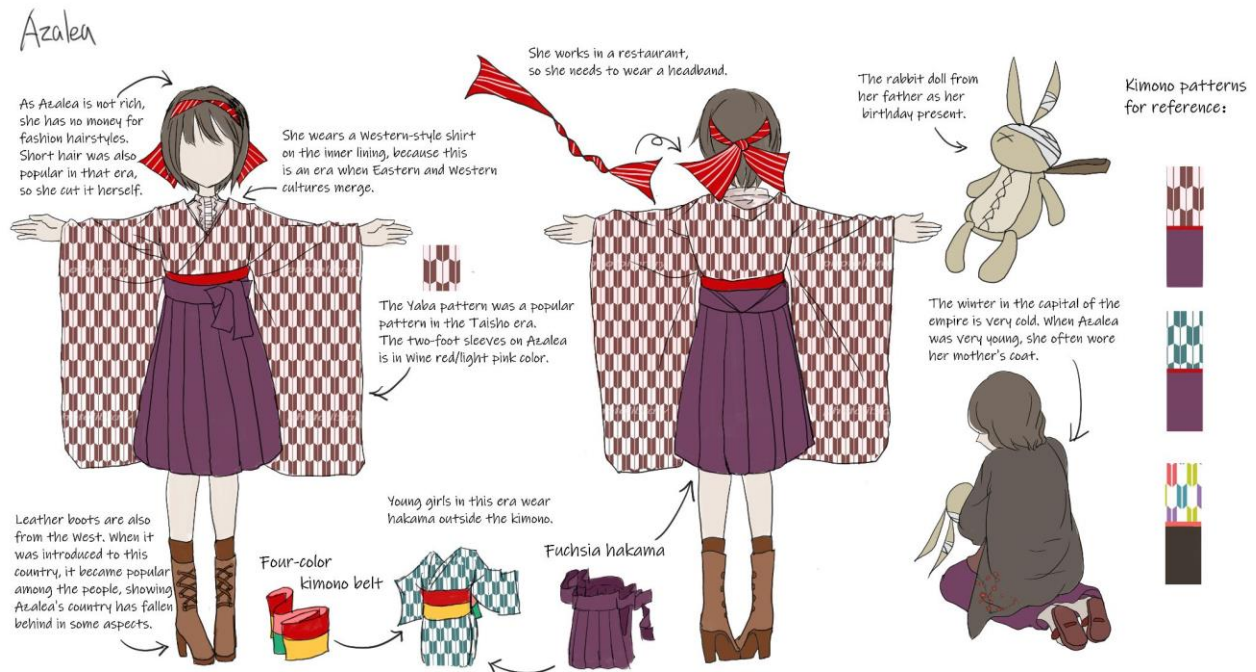
Style Frame:



Character Design:



L: Character design-Azalea (draft) R: Character Design- Azalea (1st Version)



Character Design- Azalea (Final Version)

4.2 Storyboard

Storyboard- scene 1



Storyboard- scene 2



Storyboard- scene 3

But she suddenly saw an azalea flying outside the building. So she run down the small stairs and chase after it.



The city was bombed by the planes. However when chasing the flower, the girl get out of the building and survived.

Storyboard- scene 3

She tries to find a shelter. But she falls into a cave. Some azalea flowers lead her to find the way. After she get out of the cave, she found the city was ruined.



Storyboard- scene 4



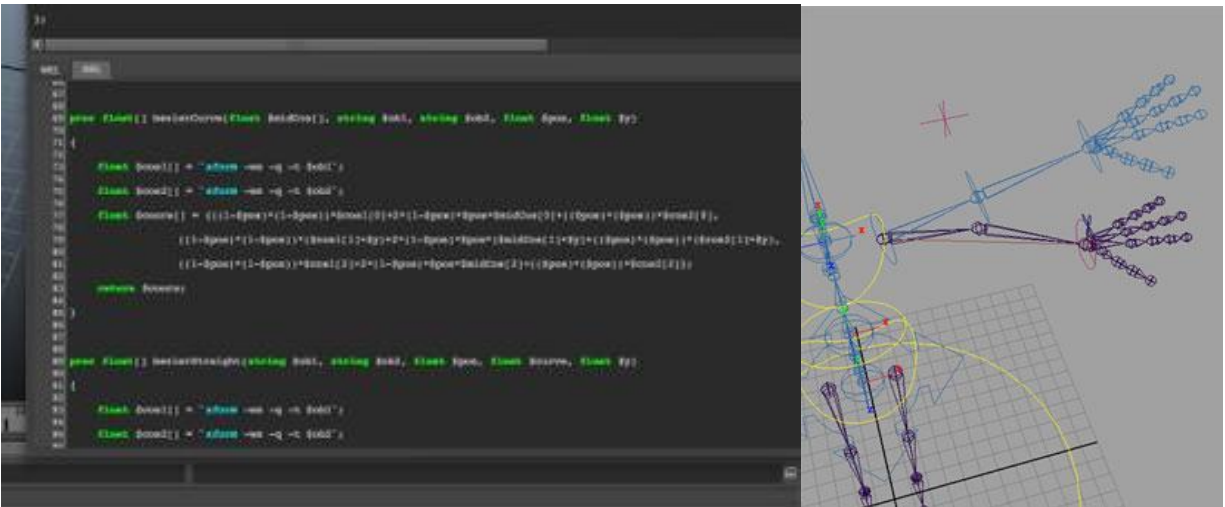
The azalea flew into the woods. Girl walked into the woods, and found a grave with her father's (who was a military officer) photograph.

5 Implementation and evaluation

5.1 Rigging

The integrated rigging of characters is the most commonly used in the process of animation production because as long as there is an animation with a character, the integrated rigging of the character must be performed. The main work involved in the process includes: creating bones, adjusting bone axes, adding IK handles, controller making, IK FK switching, constraints, skinning, Mel rigging and attribute connection, etc.

Since manual rigging requires a lot of human body structure knowledge, I use code to modify scripts to build the rigging of the main character in this project.



First of all, to create bones for the character model. The creation of bones should conform to the natures of character movement and physiological structure. Character rigging usually starts at the waist. The waist binding includes IK creation, stretching and rotation settings, global and secondary controls, etc., which can achieve the following effects. The binding of the arm mainly includes the stretching effect of the arm, the seamless switching of IK/FK, and the seamless space locking of the arm, etc., which can achieve effects such as rotating the wrist and elbow.

```

8 //
9 global proc PointCreation()
10 {
11
12     if(!window -ex CreationJoint) {deleteUI CreationJoint;}
13
14     window -title "PointCreation" -sizeable 1 -tlb 0 CreationJoint
15     rowColumnLayout -numberOfColumns 1 ;
16         text -l "";
17         text -label "===Select the point of the model.===";
18         text -label "=== UV must not overlap. ===";
19         text -l "";
20     button -label "Point creation joint" -bgc 0.5 0.65 0.5 -relief r
21
22     separator -h 5 -style "out" "a";
23
24     rowColumnLayout -numberOfColumns 1 ;
25         text -l "";
26         text -label "===Select the point of the model.===";
27         text -l "";
28     button -label "Point creation locator" -bgc 0.5 0.65 0.5 -relief r
29
30     separator -h 5 -style "out" "a";
31
32     rowColumnLayout -numberOfColumns 1 ;
33         text -l "";
34         text -label "===Select an object.===";
35         text -l "";
36     button -label "Centre creation joint" -bgc 0.5 0.65 0.5 -relief r
37
38     separator -h 5 -style "out" "a";
39
40     rowColumnLayout -numberOfColumns 1 ;
41         text -l "";
42         text -label "===Select an object.===";
43         text -l "";
44     button -label "Centre creation locator" -bgc 0.5 0.65 0.5 -relief r
45     text -label "=== WeiHero ===" -bgc 0.2 0.2 0.2 -en 1;
46
47     setParent ...:

```

The rigging of the foot makes basic settings for the foot, including adding foot stretching effect, stretching switch, and foot space locking, etc., and finally can realize the basic shape of the foot, such as heel lift, toe lift, Rotate the feet left and right, etc. Finally, define a global script command for the foot, and implement the right-click menu function.

The main principle of face rigging is the same as body binding, which includes skeleton setting, weight drawing, expression production, controller addition, secondary controller principle, and secondary expression production, etc. The face expression effect is shown in the figure below.



5.2 Material

In this short animation, all the models, materials, and textures of the objects are made by myself. In the construction of materials, I mainly use bump maps, color maps, transparency maps, normal maps, and other methods to achieve.

Bump mapping is an application of texture. It is mainly used to achieve uneven effects such as bricks and walls. Compared with general texture mapping, it does not change the color of the material itself through texture mapping, but it is to changes or perturbs the direction of its normal, and the direction of the normal is used in the light model. Changing the direction of the normal can affect the light and shade effect of the surface of the object. Therefore, bump mapping is actually a deceptive method. It does not change the position of the vertices and makes the model of the object itself bumpy, but affects the user's visual effect, making the user

think that the model is bumpy. The implementation of the bump map requires only 2 triangles, instead of a complex model that users mistakenly believe.

If we represent the vector as a row vector, we can get :

$$\begin{bmatrix} X_t & Y_t & Z_t \end{bmatrix} * \begin{bmatrix} T_x & T_y & T_z \\ B_x & B_y & B_z \\ N_x & N_y & N_z \end{bmatrix} = \begin{bmatrix} X_w & Y_w & Z_w \end{bmatrix}$$

```

},
v2f vert(appdata i)
{
    v2f o;
    o.uv = i.uv;

    o.pos = UnityObjectToClipPos(i.vertex);
    o.worldPos = mul(unity_ObjectToWorld, i.vertex); //顶点世界坐标
    o.normal = normalize(UnityObjectToWorldNormal(i.normal)); //顶点的法线
    o.tangent = normalize(mul(unity_ObjectToWorld,i.tangent).xyz); //顶点的切线
    o.bittangent = normalize(cross(o.normal, o.tangent)); //顶点的副切线
    o.light = WorldSpaceLightDir(i.vertex); //顶点的光线方向
    return o;
}
fixed4 frag(v2f i) : COLOR
{
    fixed4 texColor = tex2D(_MainTex,i.uv); //材质颜色
    float3x3 tangentTransform = float3x3(i.tangent,i.bittangent,i.normal); //世界空间-切线空间 变换
    fixed3 norm = UnpackNormal(tex2D(_NormalTex,i.uv)); //切线空间下的法线
    half3 worldNormal = normalize(mul(norm,tangentTransform)); //变换法线

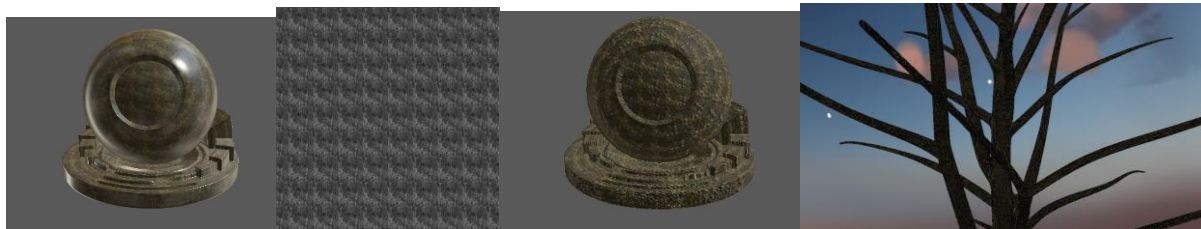
    //光照计算
    fixed3 worldViewDir = normalize(UnityWorldSpaceViewDir(i.worldPos));
    fixed3 ambi = UNITY_LIGHTMODEL_AMBIENT.xyz;
    fixed3 diff = _LightColor0.rgb * saturate(dot(worldViewDir,worldNormal));
    fixed3 lightRef = normalize(reflect(-i.light,worldNormal));
    fixed atten = LIGHT_ATTENUATION(i);
    fixed3 spec = _LightColor0.rgb * pow(saturate(dot(lightRef,worldViewDir)),_Specular)*_Gloss;
    fixed4 fragColor;
    fragColor.rgb = float3((ambi+(diff+spec)*atten)*texColor);
    fragColor.a = 1.0f;
    return fragColor;
}
ENDCG
}
}

```

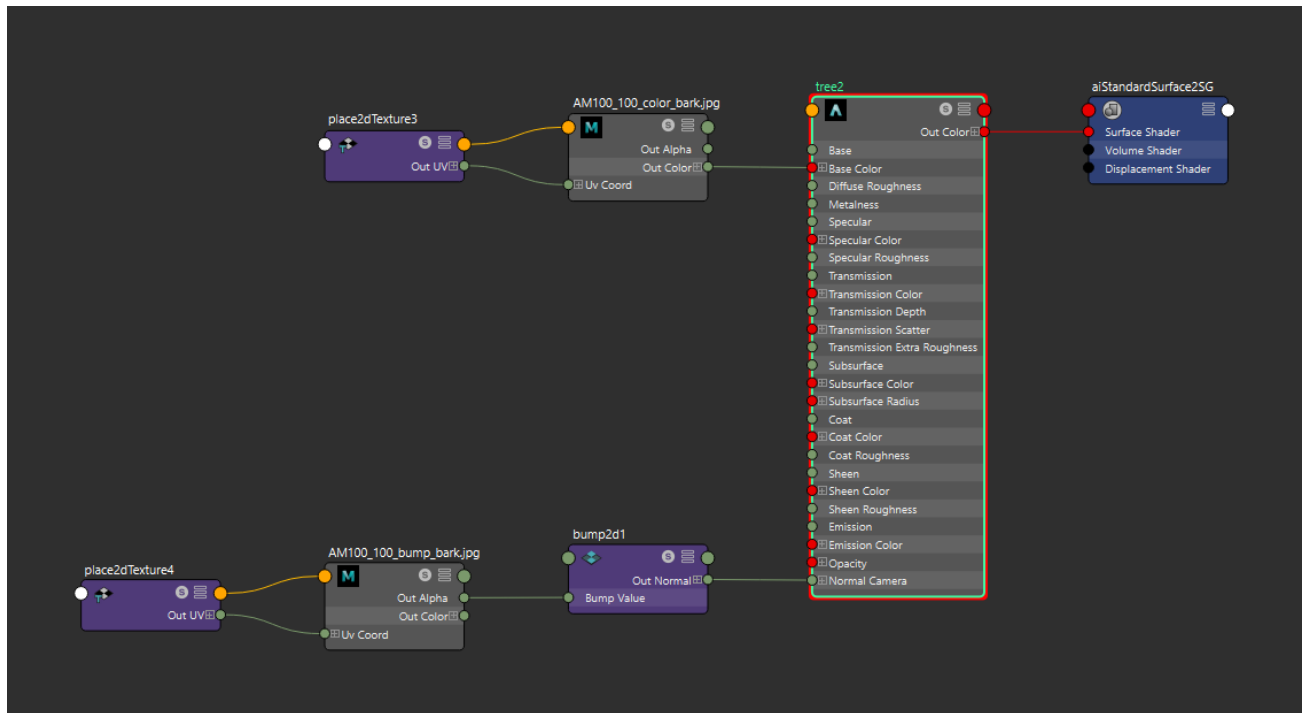
The following images show the real application in my project:



The first picture is the original object material, and the second picture is the effect after the material is given to the object (branch). Figure 3 is the color map, and Figure 4 is the effect of being pasted on the object.



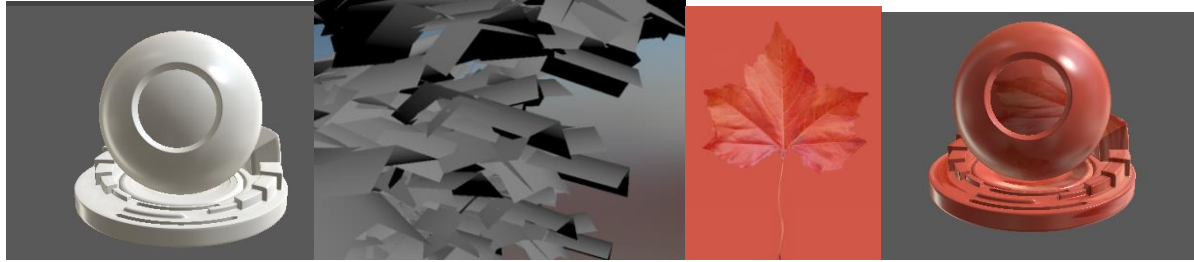
The first picture is the effect of only the color map and no bump map. The second picture is the added black and white bump map, the third picture is the material effect after adding the bump node, and the fourth picture is the effect after the final material is given to the branch. The way most bump maps are implemented is to have the normal texture store its coordinates in tangent space, in tangent space. For a vertex, its normal direction is known, and now it is necessary to construct a coordinate system, let the normal direction be the Z-axis, and then select two tangents through the point as the other two axes, one of which is called tangent, another called Bit-Tangent. The image below is the material node graph for this texture.



The following shows the final render effect of the object with this material.



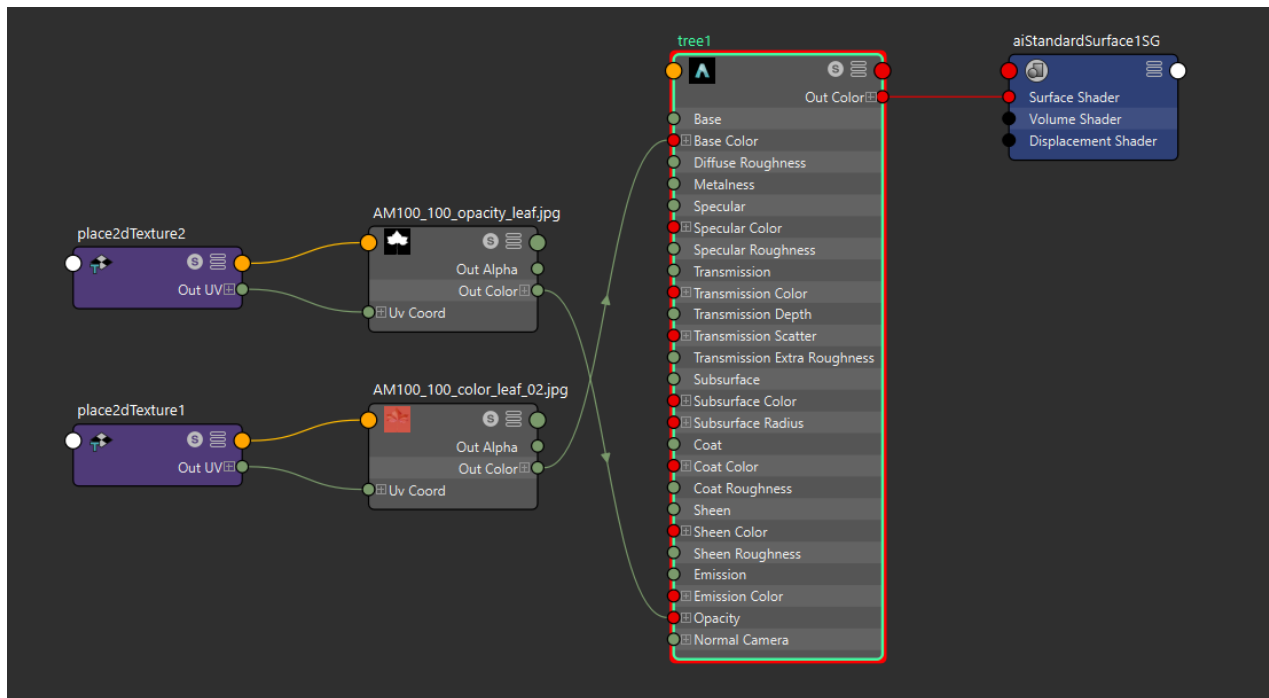
Another example is about opacity maps:



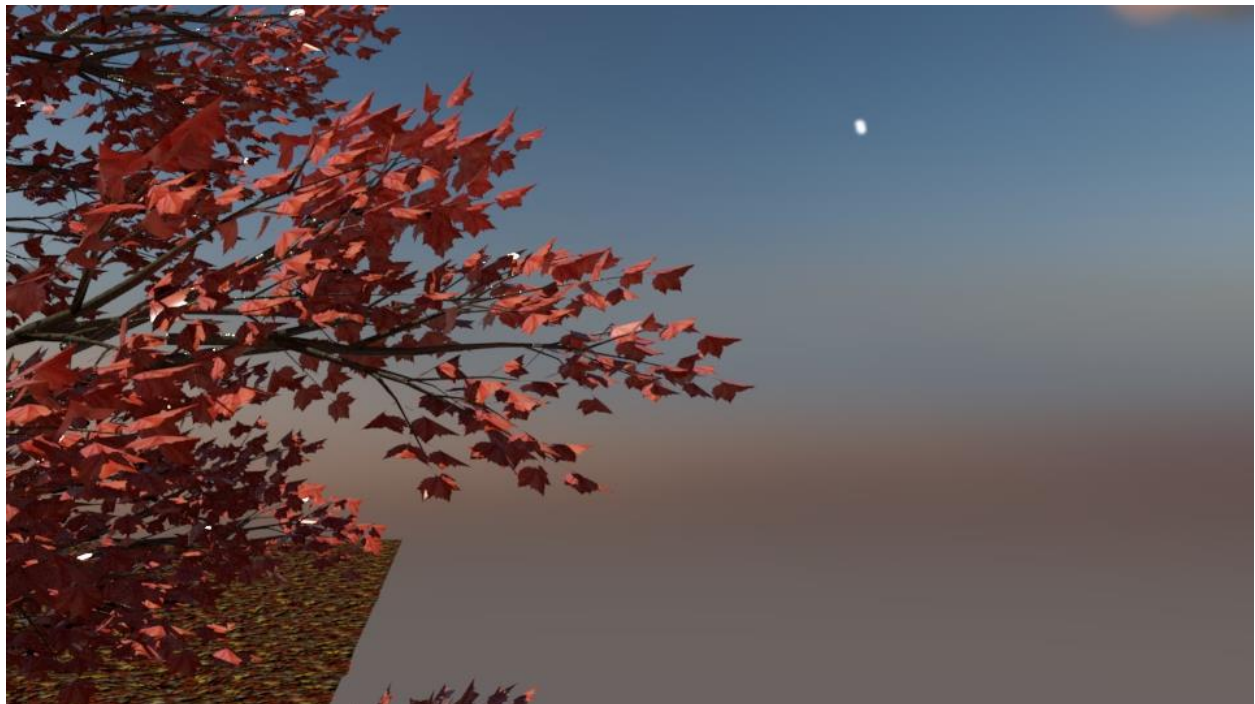
The first image shows the original object material, and the second picture is the effect after the material is adapted to the object (leaves). The leaf model is rectangular because reducing the number of faces reduces the renderer's time. The third picture is the color map of the leaves. The color map is used to create a new node through the displacement shader, and it is assigned to the original material. The shader appears as shown in Figure 4.



However, since the model of the leaf itself is rectangular, the texture will have the effect of framing a leaf in a rectangle. Therefore, add a new node to the transparency property of the material and add a transparent map, as shown in Figure 2. The material effect after adding the transparent map is shown in Figure 3 and Figure 4. The surrounding transparent parts appear to be black, but the volume of this part is ignored when rendering. The following figure is the node structure diagram of this leaf material.

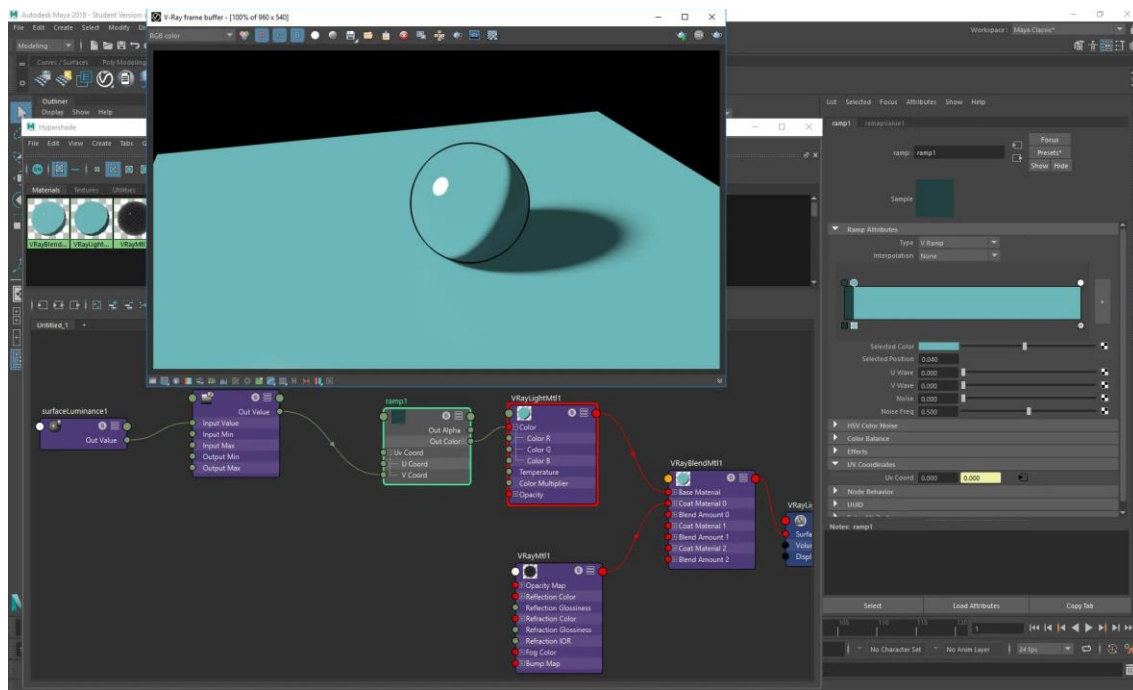


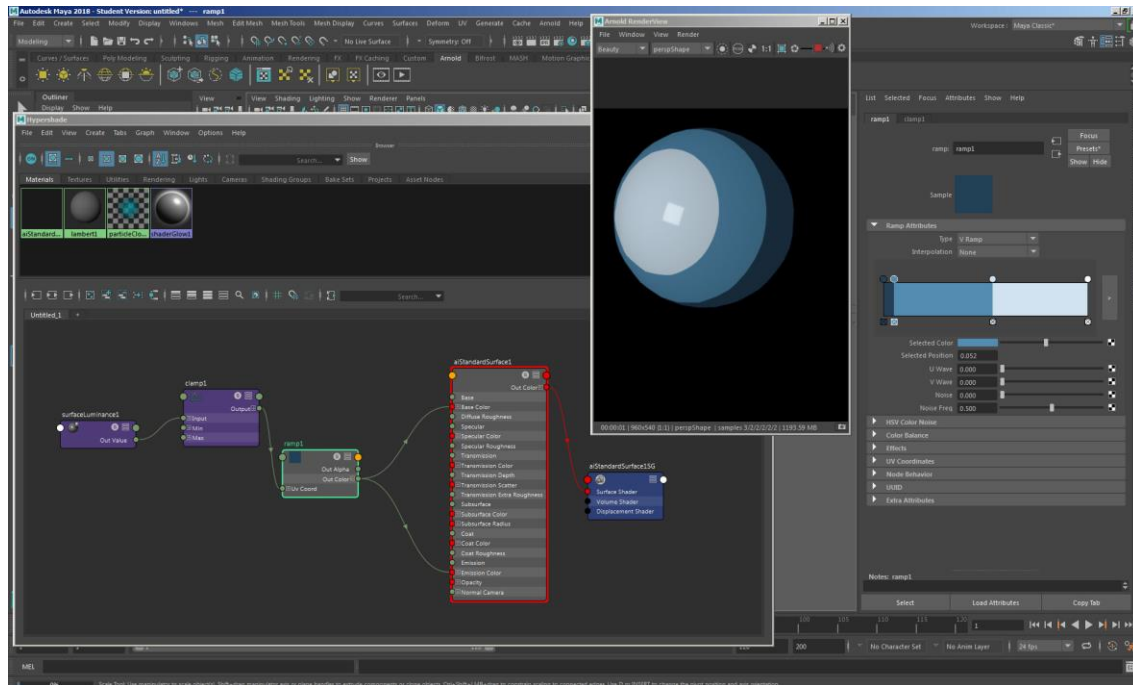
The following picture is the rendering effect.



5.3 Shader

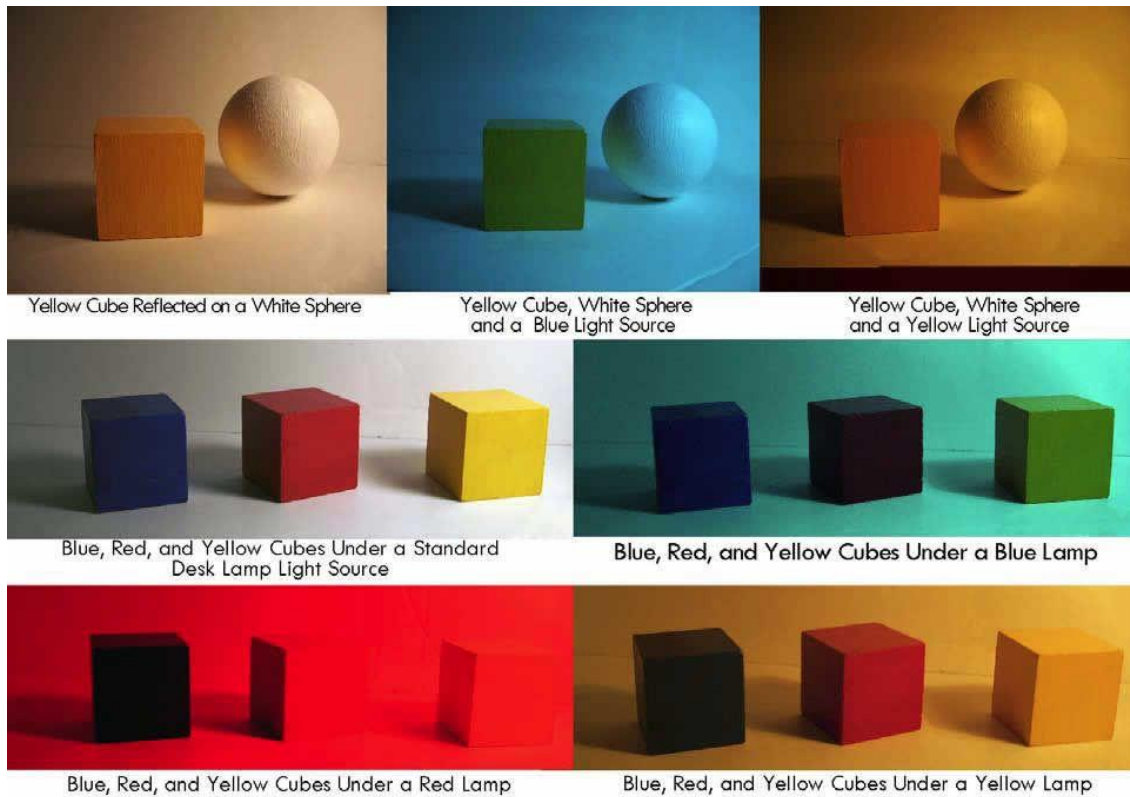
In my early research, the most preparation I did was to do research about shaders. Also, I did a lot of shader testing at semester A. The principle of the cartoon shader in maya mostly follows the following frame: add a ramp shader to the color channel and alpha channel of a common material (usually Phong). The ramp shader specifies the color A of the bright surface and the color B of the dark surface. The original base material will calculate the color value of the material after encountering the light, but the channel is passed to the ramp shader, and the ramp shader will determine it as color A or color B.





(The pictures are from the essay, my project did not use this coloring method at the end)

But there are many remaining problems in the practical operation. First, the ramp shader is not sensitive to complex lighting, especially light from multiple directions. In addition, in daily life, the color of objects will also become different under different colors of light. But the color set by the ramp shader itself is fixed, so it will look very unnatural. Moreover, since the toon shader is essentially a variant of the phong shading method, it is more suitable for the shading of metals, but it is not suitable for some less reflective materials, such as cloth and the human body.



My first render used the originaltoon render script, so I found more problems. When I finally checked the information, I found that there are three ways to make the visual think that the picture tends to be a cartoon style

1. Reduce the color levels
2. Separation of cool and warm tones
3. Hand-drawn control of light and dark areas

In fine art, it is divided into warm tones (red, yellow) and cool tones (blue, purple) according to color. In the realistic lighting calculation, only one light-dark relationship is often calculated, and then the final effect is determined by the color of the light and the object. Toon rendering, on the other hand, assigns different shades of color to light and dark sides according to the relationship

between light and shade. For example, a bright side with a warm tone and a dark side with a cool tone. After the tones are pulled apart, it further gives a cartoonish look. In hand-drawn animation, the distribution of light and shade is often not completely correct for a good picture effect. For example, the character's neck part usually appears as an obvious shadow in cartoon drawings. The result of the classical lighting calculation is very "correct" and thus lacks the hand-drawn feel of the cartoon. Lighting calculations need to be adjusted in other ways. Characters rendered in cartoons can have the best picture performance under some lighting directions. Sometimes this light direction is inconsistent with the scene light or the light direction of other characters. For the best performance of each character, it is best for each character to have its own light direction. Even when the character turns, the light turns with the character to a certain extent, so that the character has a better light and shadow performance.

Normal direction control: There are two methods for normal control. One is to directly edit the normal to achieve the desired lighting result. One is to create a smooth simple model and then transfer its normals to complex objects to optimize shadows. Maya has its own normal transfer function, and 3ds Max can implement the normal transfer function through the plug-in Noors Normal Thief.

First of all, we implement a lighting effect with a clear boundary between light and dark and support setting the color of light and dark areas separately, setting the color of the dark side as a cool tone, and distinguishing it from the tone of the bright side.

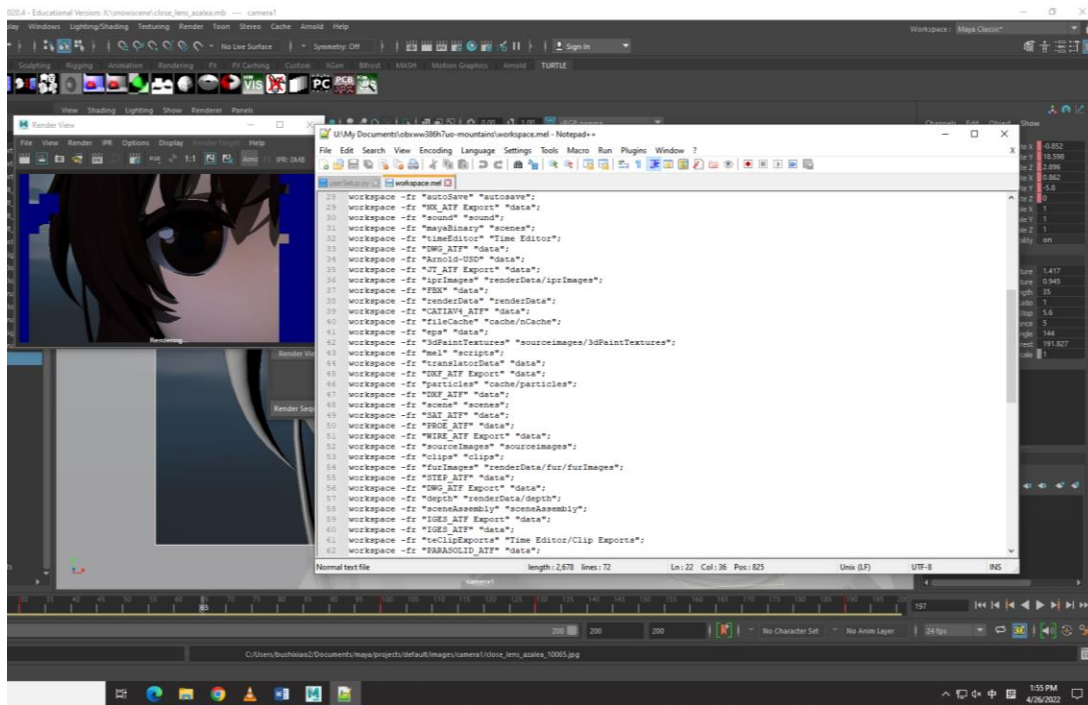

```
*shader - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
userSetup.py workspace.mel shader

1 Shader "Unlit/CelRender"
2 {
3   Properties
4   {
5     _MainTex ("MainTex", 2D) = "white" {}
6     _MainColor ("Main Color", Color) = (1,1,1)
7     _ShadowColor ("Shadow Color", Color) = (0.7, 0.7, 0.8)
8     _ShadowRange ("Shadow Range", Range(0, 1)) = 0.5
9     _ShadowSmooth ("Shadow Smooth", Range(0, 1)) = 0.2
10
11   [Space(10)]
12   OutlineWidth ("Outline Width", Range(0.01, 2)) = 0.24
13   _OutLineColor ("OutLine Color", Color) = (0.5,0.5,0.5,1)
14   }
15   SubShader
16   {
17     Tags { "RenderType"="Opaque" }
18
19     pass
20     {
21       Tags { "LightMode"="ForwardBase" }
22
23       Cull Back
24
25       CGPROGRAM
26       #pragma vertex vert
27       #pragma fragment frag
28
29       #include "UnityCG.cginc"
30       #include "Lighting.cginc"
31       #include "AutoLight.cginc"
32
33       sampler2D _MainTex;
34       float4 _MainTex_ST;
35       half3 _MainColor;
```

The built-in scripts in Maya are all py files, so they can be edited with any python editor.



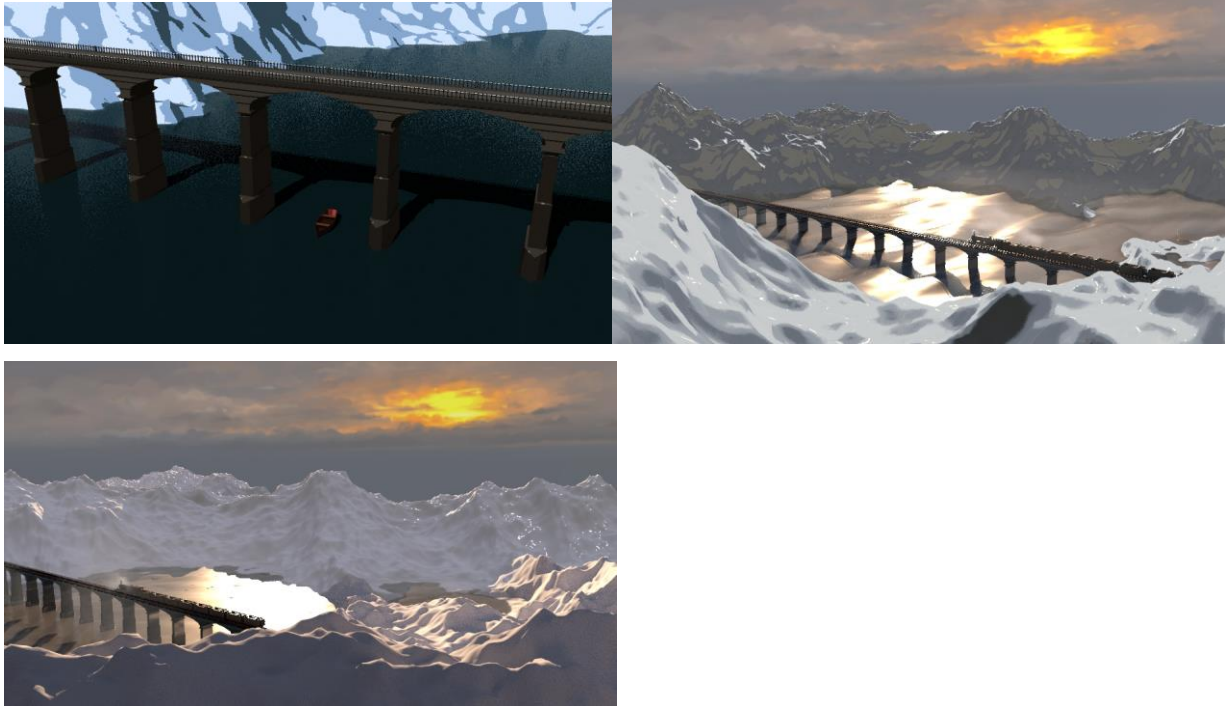
Left is the gray model of the original character, and on the right is the rendering output of the toon shader.



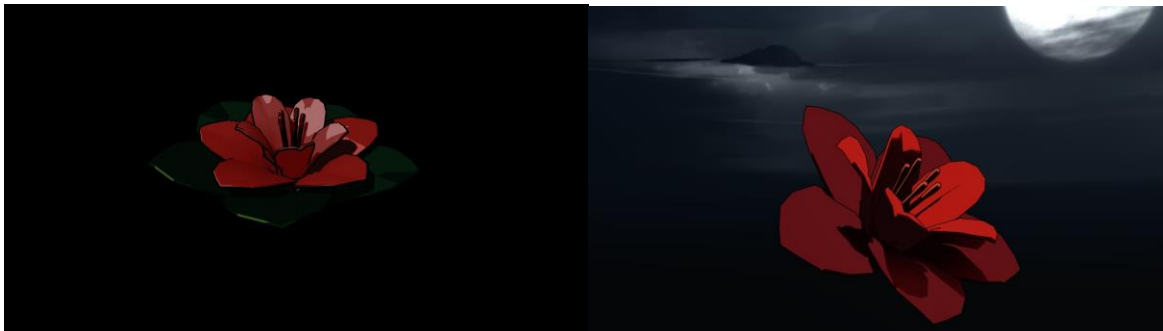
But after that, I found out that toon shader is not good for shading clothes. As shown below, the left used a toon shader for rendering, and the clothes on the right used the material of the AI standard surface.



The image below is an attempt at three different ways of shading. The first one is a renderer using Maya software, using a two-color shader, the second and third are using the arnold renderer, but the second one set an external ramp shader, so it looks more like drawn, but Highlights and shadows are therefore somewhat unnatural. The third one is shaded using an AI standard surface.



The figure below is also two different render effects from two rendering scripts, rendering the same model, but both in the style of cartoon rendering.



In addition, in the first version of the toon shader some bugs occur, for example, the shader used will have a glowing effect in dark scenes, as shown in the rendered images below.



So I started to rework the shader script. The idea is still to connect the alpha channel and color channel of the basic material to external shaders, but the shading method is changed from a single ramp shader to a map. Another approach is to control the color level and light and dark boundaries by sampling the Ramp map. It can be seen as using the result of standard lighting as UV, sampling a map used as a color map, and controlling the result of lighting calculations through this map.



Left: old shader effect. Right: New shader effects and ten main materials on the main character. Ramp maps make it easier to define multiple color levels, but it is necessary to make unique maps one by one. Since there is only one main character, the textures are not complicated to make.

```

ad++
h View Encoding Language Settings Tools Macro Run Plugins Window ?
workspace.mel shader ramp

_OutLineColor ("OutLine Color", Color) = (0.5,0.5,0.5,1)

bShader

Pass
{
    Tags { "LightMode"="ForwardBase" }

    CGPROGRAM
        #pragma vertex vert
        #pragma fragment frag
        #pragma multi_compile_fwdbase

        #include "UnityCG.cginc"
        #include "Lighting.cginc"
        #include "AutoLight.cginc"

        sampler2D _MainTex;
        float4 _MainTex_ST;
        sampler2D _ImlTex;
        float4 _ImlTex_ST;

        half3 _MainColor;
        half3 _ShadowColor;
        half _ShadowSmooth;
        half _ShadowRange;

        half3 _SpecularColor;
        half _SpecularRange;
        half _SpecularMulti;
        half _SpecularGloss;

        struct a2v
        {
            float4 vertex : POSITION;
            float2 uv : TEXCOORD0;
            float3 normal : NORMAL;
        };

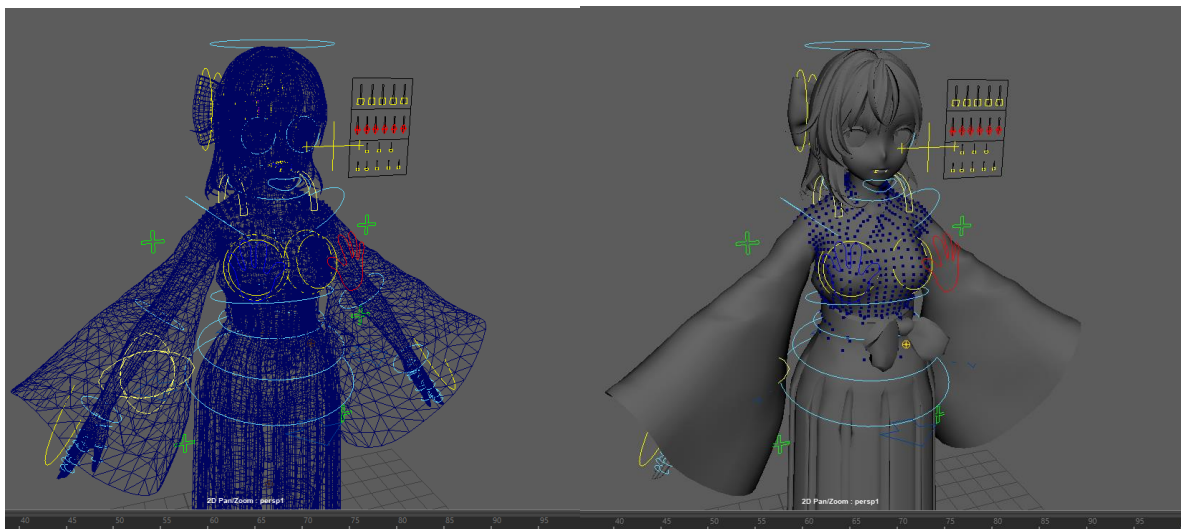
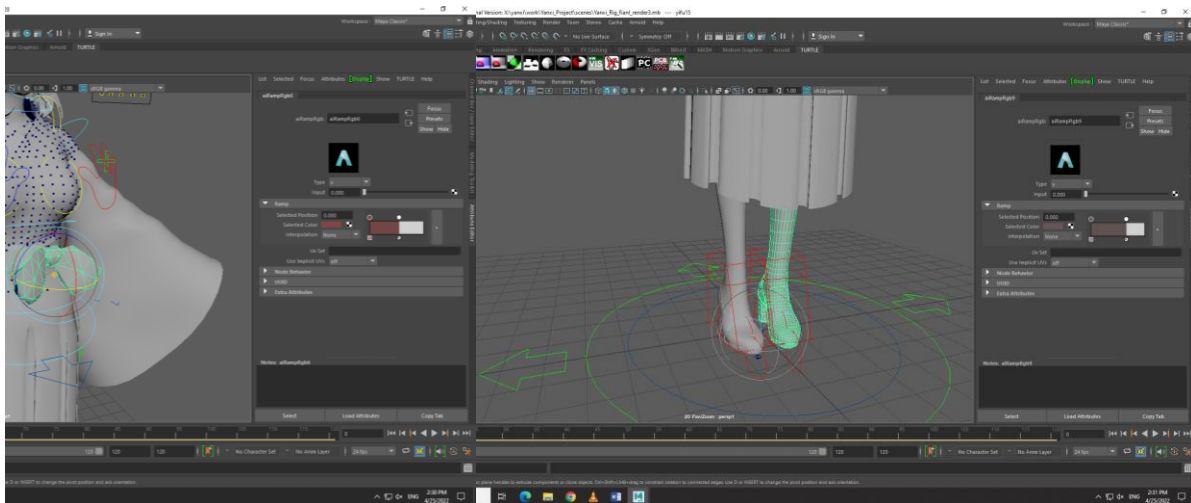
        struct v2f
        {
            float4 pos : SV_POSITION;
            float2 uv : TEXCOORD0;
            float3 worldNormal : TEXCOORD1;
            float3 worldPos : TEXCOORD2;
        };

        v2f vert (a2v v)
        {
            v2f o = (v2f)0;
            o.pos = UnityObjectToClipPos(v.vertex);
            o.uv = TRANSFORM_TEX(v.uv, _MainTex);
            o.worldNormal = UnityObjectToWorldNormal(v.normal);
            o.worldPos = mul(unity_ObjectToWorld, v.vertex).xyz;
            return o;
        }
    }

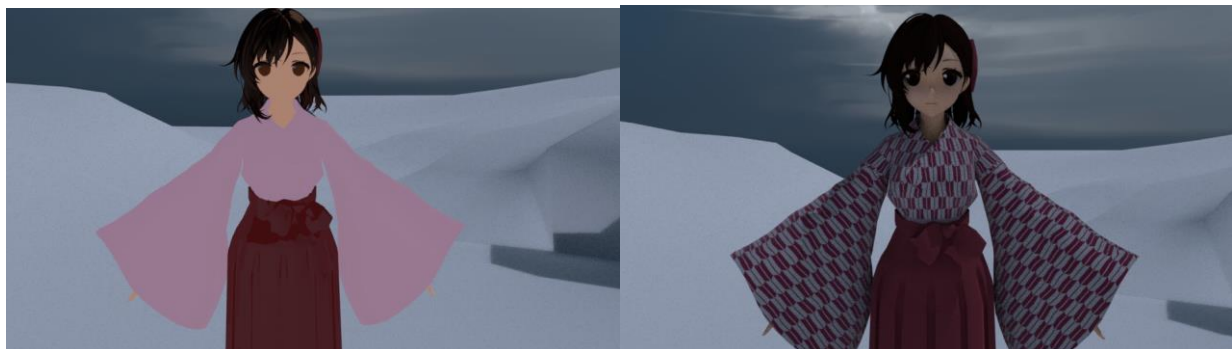
```

Define an external node: ramp map



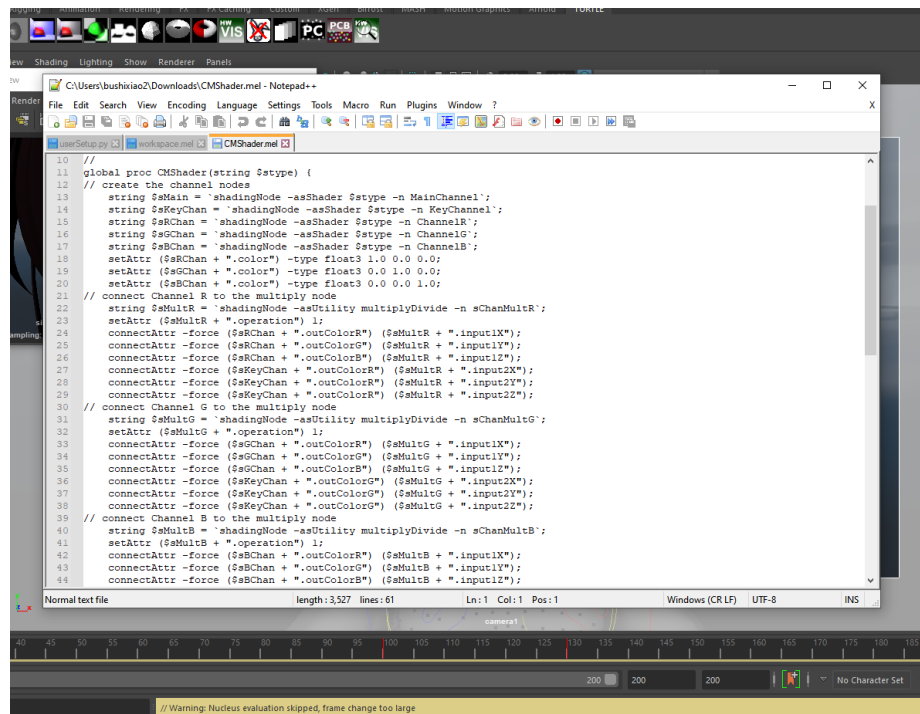


The images above are the original model and the ramp color settings of shaders.



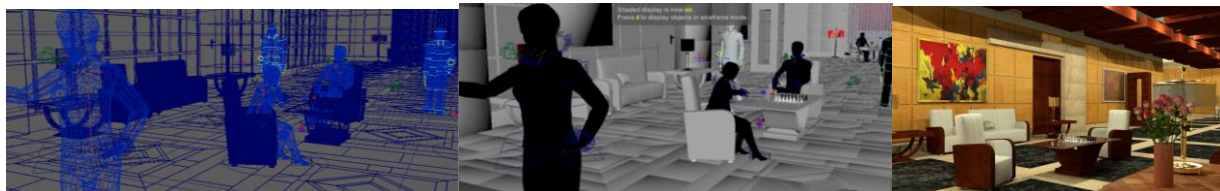
The left side of the above image is the effect of the original shader in the scene, and the right side is the effect of the new shader in the scene. Obviously, the skin tones on the right will change

with the lighting and the outside color of the scene, so that there is a connection between the character and the scene without making the character stand out.

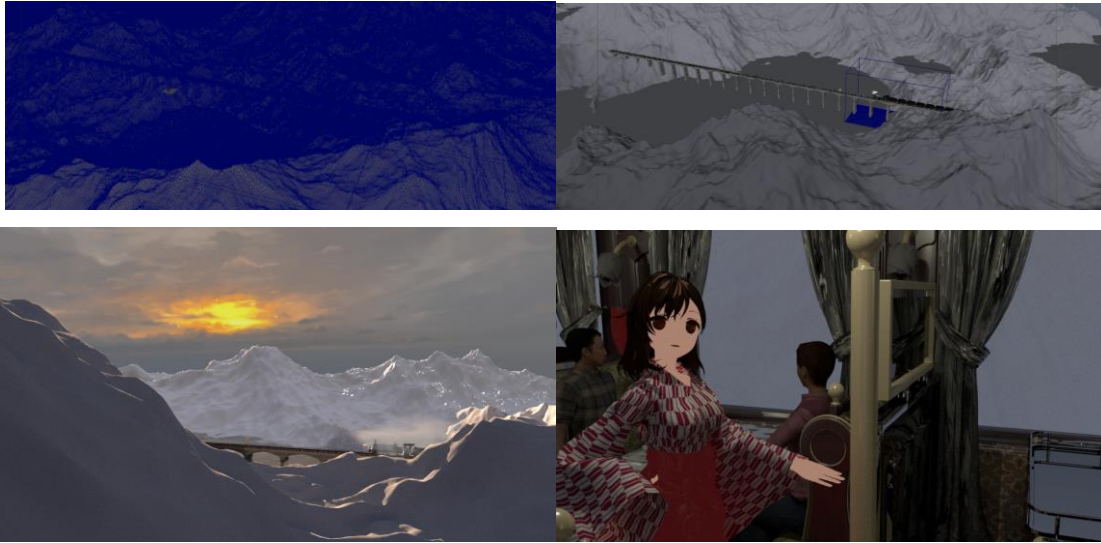


```
10 //
11 global proc CMShader(string $stype) {
12 // create the channel nodes
13 string $sMain = 'shadingNode -asShader $stype -n MainChannel';
14 string $sKeyChan = 'shadingNode -asShader $stype -n KeyChannel';
15 string $sRChan = 'shadingNode -asShader $stype -n ChannelR';
16 string $sGChan = 'shadingNode -asShader $stype -n ChannelG';
17 string $sBChan = 'shadingNode -asShader $stype -n ChannelB';
18 setAttr ($sMain + ".color") -type float3 1.0 0.0 0.0;
19 setAttr ($sRChan + ".color") -type float3 0.0 1.0 0.0;
20 setAttr ($sGChan + ".color") -type float3 0.0 0.0 1.0;
21 // connect Channel R to the multiply node
22 string $sMultR = 'shadingNode -asUtility multiplyDivide -n sChanMultR';
23 setAttr ($sMultR + ".operation") 1;
24 connectAttr -force ($sRChan + ".outColorR") ($sMultR + ".input1X");
25 connectAttr -force ($sRChan + ".outColorG") ($sMultR + ".input1Y");
26 connectAttr -force ($sRChan + ".outColorB") ($sMultR + ".input1Z");
27 connectAttr -force ($sKeyChan + ".outColorR") ($sMultR + ".input2X");
28 connectAttr -force ($sKeyChan + ".outColorR") ($sMultR + ".input2Y");
29 connectAttr -force ($sKeyChan + ".outColorR") ($sMultR + ".input2Z");
30 // connect Channel G to the multiply node
31 string $sMultG = 'shadingNode -asUtility multiplyDivide -n sChanMultG';
32 setAttr ($sMultG + ".operation") 1;
33 connectAttr -force ($sGChan + ".outColorR") ($sMultG + ".input1X");
34 connectAttr -force ($sGChan + ".outColorG") ($sMultG + ".input1Y");
35 connectAttr -force ($sGChan + ".outColorB") ($sMultG + ".input1Z");
36 connectAttr -force ($sKeyChan + ".outColorG") ($sMultG + ".input2X");
37 connectAttr -force ($sKeyChan + ".outColorG") ($sMultG + ".input2Y");
38 connectAttr -force ($sKeyChan + ".outColorG") ($sMultG + ".input2Z");
39 // connect Channel B to the multiply node
40 string $sMultB = 'shadingNode -asUtility multiplyDivide -n sChanMultB';
41 setAttr ($sMultB + ".operation") 1;
42 connectAttr -force ($sBChan + ".outColorR") ($sMultB + ".input1X");
43 connectAttr -force ($sBChan + ".outColorG") ($sMultB + ".input1Y");
44 connectAttr -force ($sBChan + ".outColorB") ($sMultB + ".input1Z");
```

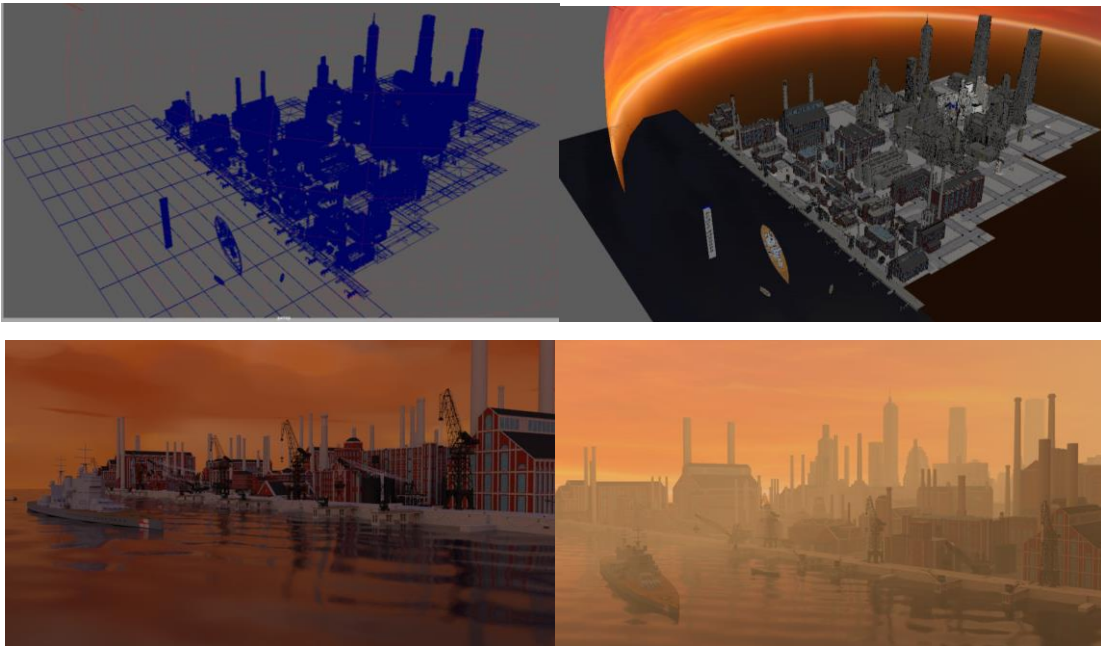
5.4 Other 3D scene models



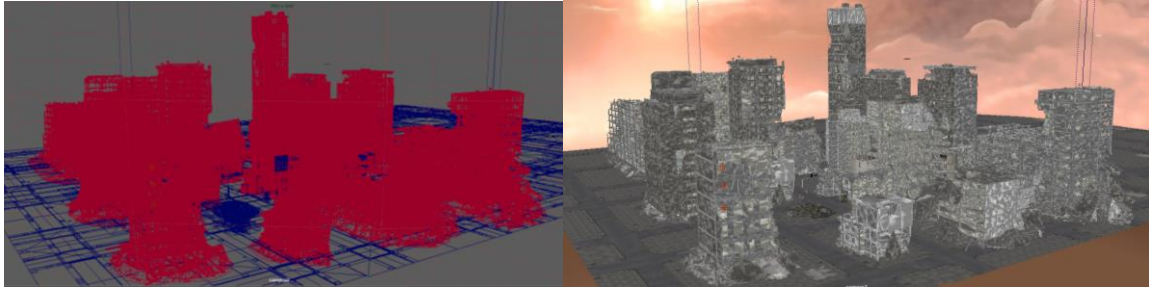
Scene: in the hall



Scene: The heroine is on the train, the train goes through the snowy mountains



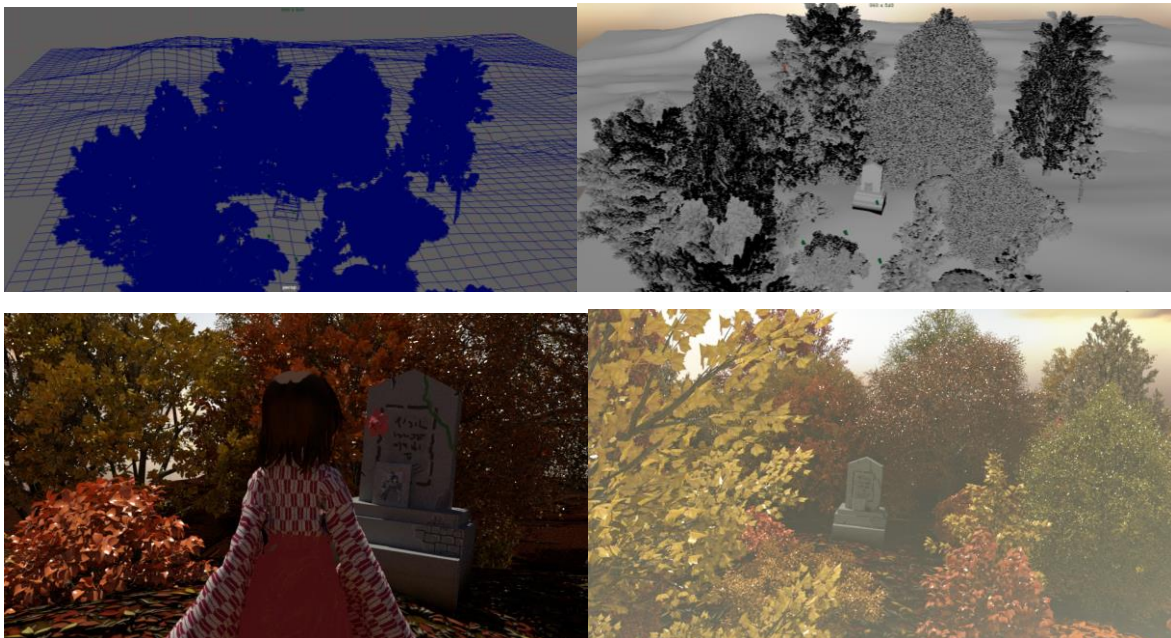
Scene: the city called Peace Harbor



Scene: The city that was bombed to ruins



Scene: Abandoned bomb shelter

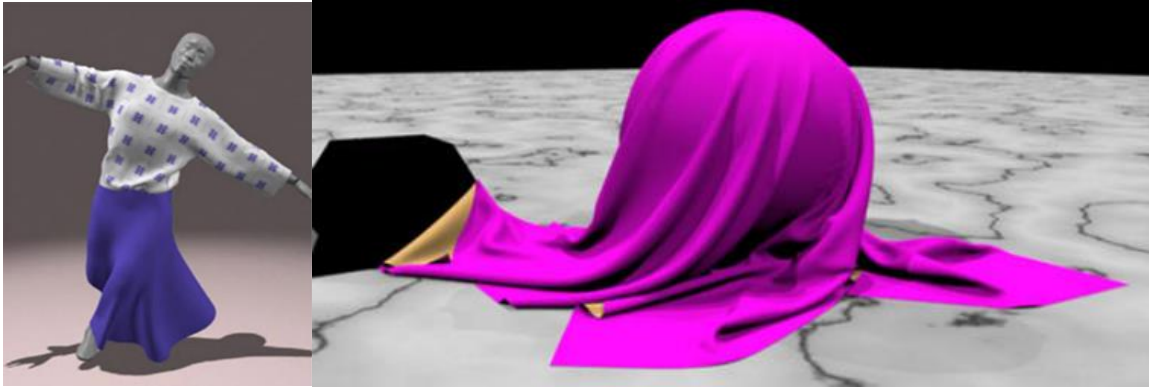


Scene: woods

5.5 Cloth Simulation

With the development of the tech and the improvement of the hardware performance, physics engines are used by more and more mobile games. The objects processed by the physics engine can be mainly divided into two parts: Rigid Bodies and Soft Bodies. Rigid Bodies are mainly used in systems such as physical collision, fragmentation, and ragdolls, and the most important application representative of Soft Bodies is the cloth system. In the physics engine, the system that simulates the cloth effect through physical calculation can be called the cloth system. Although it is called a cloth system, it is not only used to express the cloth effect of the character's clothes, but can also be applied to hair, pendants, etc.

The research on cloth simulation in the field of computer graphics can be traced back to the 1980s. At the earliest, the research on the cloth system in the academic world was definitely not used for real-time calculation of cloth, but offline rendering, or even just to generate a static image. For example, Terzopoulos' paper in 1987 proposed a method to treat the cloth as a rectangular grid, and then use a semi-implicit integration method to update its position, using elasticity theory to simulate animation effects. Breen proposed a new perspective in 1994, that is, a Particle-based approach, using the Kawabata measuring system for simulating fiber fabrics to simulate the running effect of cloth. In their 1998 paper, Baraff and Witkin proposed a scheme to allow larger time steps while maintaining a stable cloth simulation, which has also been gradually improved to a faster method. After that, algorithm optimization based on data structures such as AABB tree, collision with rigid bodies, and collision calculation of cloth itself were gradually studied and corresponding solutions were proposed. Cloth simulation has gradually developed from an academic research direction to a possible Industrial technology applied to film and television production, and then applied to the field of games that require real-time computing. The following figures show the cloth simulations shown by several important papers of the year.



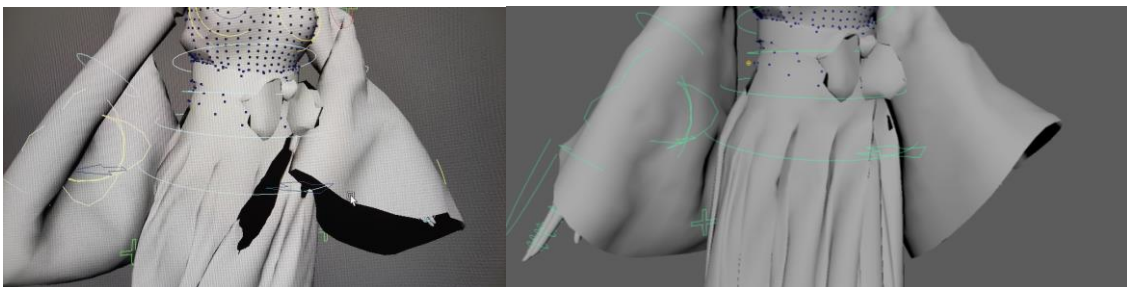
Cloth effects are simulated using large time steps by Baraff and Witkin, 1998

Cloth simulation technology is roughly composed of four parts: cloth modeling, numerical calculation, collision processing, and rendering display. As can be seen from the previous article, in the research and development of more than 30 years, researchers in various countries have proposed a variety of different methods for these technologies. Among them, cloth modeling is the most important part, because it affects and even determines the subsequent Implementation mechanism for numerical computation and collision handling. There are many modeling methods that have been proposed and applied, such as the earlier catenary linear model, the cloth deformation simulation method of pure set transformation, the physics-based elastic deformation model, the particle system model, etc., and even texture-based geometry generation. Folding method. The models implemented and integrated into modern game engines are usually the mass-spring model based on the physics method, which has the characteristics of a simple model and high computational efficiency.

In the mass-spring model, the cloth is modeled as particles on a mesh, which are connected by spring dampers. Each spring connects two particles and generates a force based on the particle's position and velocity. Particles can be affected by gravity, and springs can be set to different types, such as stretch springs, shear springs, and bend springs.

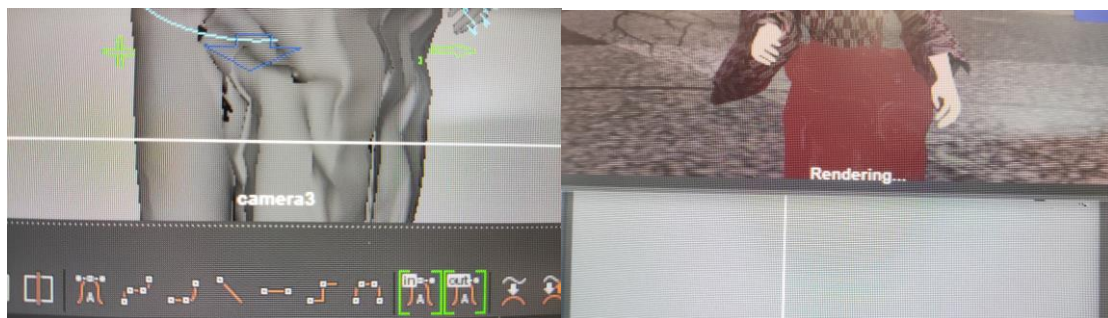


In this project, a mass-spring model is used to simulate the cloth, and a lot of decomposition calculation time is spent on the main character, since she wears a long skirt and long sleeves, and has a bow on his head. Also, because the clothes are very long, the number of collision iterations is adjusted very high to avoid bugs, thus greatly increasing the rendering time.



In the Quality Settings area, set Max Self Collide Iterations to 20. The Max Self Collision Iterations value specifies the maximum number of self-collision-related nCloth calculations performed per substep. Increasing this value causes Maya to recognize more self-collisions, simulating more realistic cloth, but at a slower rate. Set Collision Last Threshold to 1.0.

Increasing the Collide Last Threshold helps prevent nCloth vertices from passing through colliding objects. Enabling Trapped Check and Self Trapped Check causes Maya to monitor collision intersections. When the points cross each other, Maya tries to push them away. This action prevents the skirt from clumping. Set pushout to 0.280. Setting Push Out applies a force that pushes intersecting or penetrating objects to the closest point on the surface of the current nCloth object. Set the push-out radius to 5.0. Setting the Push Out Radius specifies the maximum distance from the surface of the current nCloth object affected by the Push Out property. On the left is the result of the operation with a small number of iterations. When the protagonist moves, the skirt has holes. On the right is the number of iterations increased. There are no holes in the skirt at this time, but the calculation time and memory usage required are much more than the previous one.



In addition, there are many cloth calculation problems, which cannot be solved temporarily due to hardware limitations, but this part can be solved by repairing the rendering images. The left side of the above image is the bug that appeared in the rendering process, and the right image is the result after repairing according to the principle of similar color parameters.

6 Conclusion: a critical evaluation of the project

6.1 limitations and future developments

The first problem is the material of the scene model, because there are some places where materials and texture maps are used for convenience (such as the text on the store sign), so it may not look coordinated together with the cartoon-like objects using the toon shader. Therefore, I will continue to improve the scene and improve the material in the follow-up. Another problem is that there are too many faces for some objects. For example, in a snow-capped scene, twelve pictures need to be rendered for one and a half hours due to too many faces. Therefore, it is necessary to continuously reduce the number of faces, but this must be done without changing the rendering quality.

Also, there are still issues with cloth simulation and human rigging. Part of the reason is that due to hardware constraints, it is impossible to perform film-level cloth calculations. The current number of cloth collision iterations is around 5-20, and the computer cannot run after more than 20 times. So with better hardware conditions, there will be fewer holes in the cloth calculation. Another reason is the limitation of existing algorithms, lack of automatic regulation of AI, but only simple physical calculations. If I can do other 3D cloth simulation software, maybe the effect will be improved.

Also, another issue is that I don't currently use motion capture. So all actions are manually set keyframes by hand. If motion capture can be applied in the future, the efficiency will be much improved, and the character's movements will not be so blunt.

6.2 Summary

Overall, this project was very tough for me. First of all, I can guarantee that this project is all done by myself, unlike many final year students who hire juniors and even online programmers

to help them do final year projects. I think my project has more code content than games because animation is a field that programmers rarely set foot in than games, so there is no ready-made code for me to copy on the Internet. When making games, they can often find ready-made game templates, and after improving the code, they can become the so-called final year project of their own.

Secondly, all the models are made by myself. Like each tree in the forest scene, there are leaves, tree trunks, and branches, each with unique textures, including bump mapping, transparency map, as well as color mapping, normal mapping, and so on. There are a hundred trees, and thousands of textures are all coded and applied by me. So it's a huge amount of work.

In addition, I also did much other research than CG for this project. When I made the clothes of the characters, I almost learned a whole clothing design software for the creation of this model, and even the cutting, sewing, and so on of the 3D model of the clothes.

Overall, I think it's been a valuable experience, but for me, the greatest thing about this project is that I made this complete animation. Since this animation was originally from a portfolio I applied to the School of Creative Media at the City University of Hong Kong four years ago. Of course, the script written at that time was a complete movie, which was much more complicated than the current animation, but it is impossible for me to recover the complete script with my current technical level. And I put a lot of effort into simplifying the script. This work made me see the past of the historical script I wrote when I was 17 years old.

Grateful to my two supervisors for their help here.

6.3 Other Links

Fast Track Video (shows the production process):

<https://youtu.be/lkQl6BDIV2o>

Draft version Video (a test rendering):

<https://youtu.be/F1lFfcqgHsA>

Final Year Project Blog Folder:

<https://drive.google.com/drive/folders/1UGKfVtyCHD-SWgQ-lgAme1XZ5Sh2NkPS?usp=sharing>

7 Reference List

Mao Hui. (2009). MAYA particle and fluid special effects processing (Master's thesis, Shanghai Jiaotong University).

Baraff, D., & Witkin, A. (1998, July). Large steps in cloth simulation. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques* (pp. 43-54).

Terzopoulos, D., Witkin, A. P., & Kass, M. (1987, July). Energy Constraints on Deformable Models: Recovering Shape and Non-Rigid Motion. In *AAAI* (pp. 755-760).

McNamara, A. (2021, August). An Introduction to Python Scripting in Autodesk Maya. In *ACM SIGGRAPH 2021 Educators Forum* (pp. 1-53).

Howe, W. (2011, March). Scripting Shaders in Maya A Color-Mapped Shader. We show. <http://www.weshowe.com/dlds/MappedShader.html>

Whited, B., Daniels, E., Kaschalk, M., Osborne, P., & Odermatt, K. (2012). Computer-assisted animation of line and paint in Disney's Paperman. In *ACM SIGGRAPH 2012 Talks* (pp. 1-1).

Anjyo, K. I., & Hiramitsu, K. (2003). Stylized highlights for cartoon rendering and animation. *IEEE Computer Graphics and Applications*, 23(4), 54-61.

Thacker, J. (2021, March 7). *Download a free anime-style Maya character rig: CG Channel*. CG Channel | Community for Entertainment Artists. <http://www.cgchannel.com/2021/03/download-a-free-anime-style-maya-character-rig/>.

MouseComputer2010. (2019, August 27). ヨルシカ 「ノーチラス」 *MV* メイキングムービー *with DAIV* / マウスコンピューター. YouTube.
<https://www.youtube.com/watch?v=F06jSFsIOu4>. MORIE Inc. Gumroad. (n.d.).
https://gumroad.com/morieinc?sort=page_layout.

Heliofant. (2013, September 25). *I, Pet Goat II*. YouTube.
<https://www.youtube.com/watch?v=65xLByzT1l0>.

T-POCKET. (2017, January 4). *[60fps Full-cast] 1925 - 初音ミク リンレン ルカ KAITO MEIKO Miku Rin Len Luka DIVA English Romaji subtitles*.
<https://www.youtube.com/watch?v=ZWFHtMxyymw&t=136s>

黒うさ. (2014, December 2). 「*M V*」 千本桜 *WhiteFlame feat 初音ミク*.
YouTube. <https://www.youtube.com/watch?v=shs0rAiwsGQ&list=RD-vwCiR2oqcE&index=5>

Block, Bruce. (2001). *The Visual Story: Seeing the structure of Film, TV, and New Media*. MA, USA: Focal Press