Assignment 2 - Technical Design Document

Live Document Link:
FIT2097-xhee0044-TechnicalDesignDocument.

Xun He
30451248

Contents

Project Overview

Core Mechanic

Super Power Football is a third person game. Multiple players can use different super powers to push the football into the goal. It is similar to Lúcioball in Overwatch. The main goal is to push the football into the other football goal. Ultra-high mobility allows you to experience more intense confrontations than the other football games. Therefore, all core mechanics interact with this goal. Players

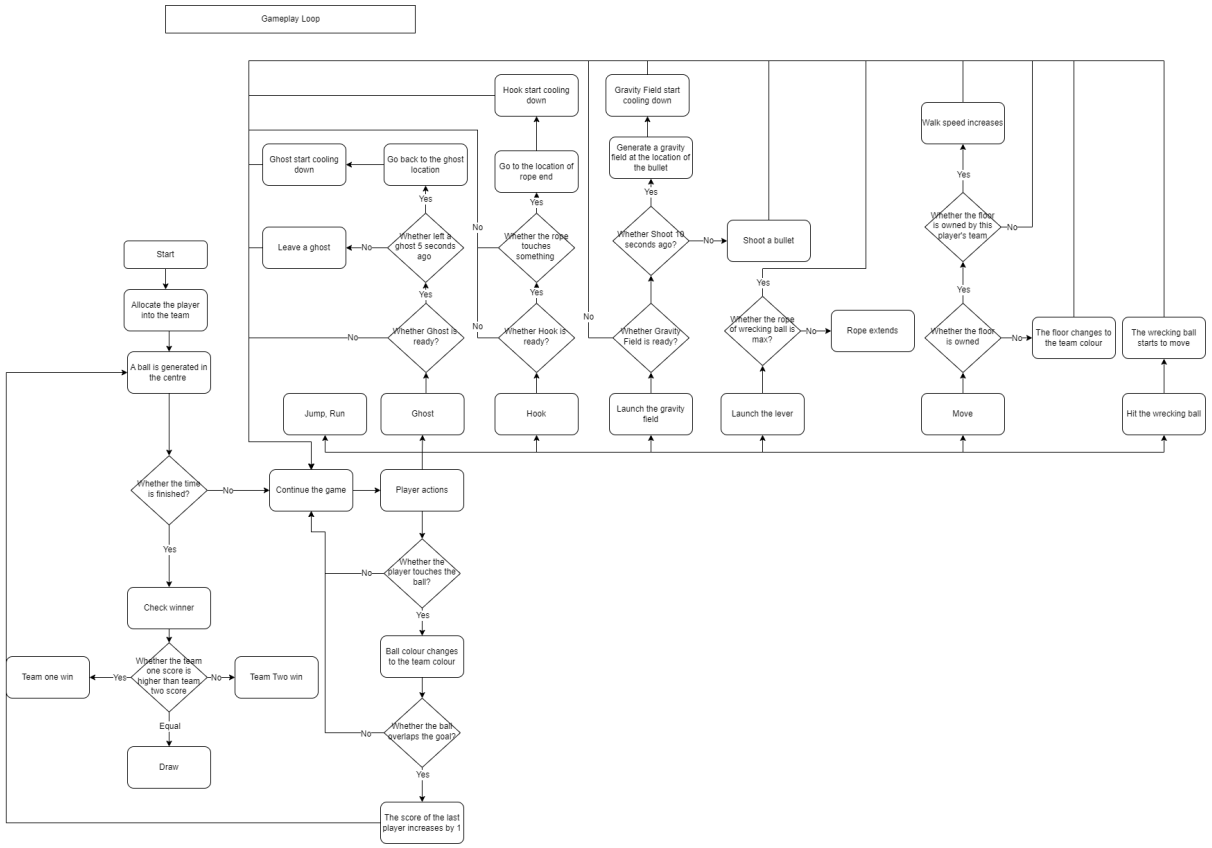
- When players press the left mouse button, they can use ForcePush on the ball. The ball will be pushed away with great force.
- When players press Q on the keyboard, they can launch a rope, then he will be pulled towards the end of the rope. This mechanic can help players keep up with rivals when their rivals pass them with the ball.
- When players press R on the keyboard, they can leave a ghost in their current positions. Then they can press R again in 5 seconds to go back to where they left the ghost. Of course, players can leave the ghost anytime and anywhere, for example, while hooking or dropping from the sky.
- When players press E on the keyboard, they can shoot a bullet. When they press E again or the bullet reaches its farthest point, there will be a gravity field generated in its position. The field will restrict the movement of enemies around it and pull them toward the centre. In the field, hooking will fail. Therefore, placing the ghost in advance is a good way to escape the field.

Target Platform

The target platform is Pc. And I will focus on binding mechanics to the keyboard and mouse rather than the gamepad. In Super Power Football, most mechanics need the second press and collimation. Through my game experience, those two points on the gamepad don't work very well with the gamepad unless the game has good Aim Assist. Therefore, on the console platform that primarily supports gamepads, Super Power Football does not make for a good gaming experience. Also, this reason can be applied to the mobile platform that uses the virtual joystick. On the PC platform, the mouse can help the player collimate accurately to hook and pressing one button twice on the keyboard is easy to launch the field or return the ghost.

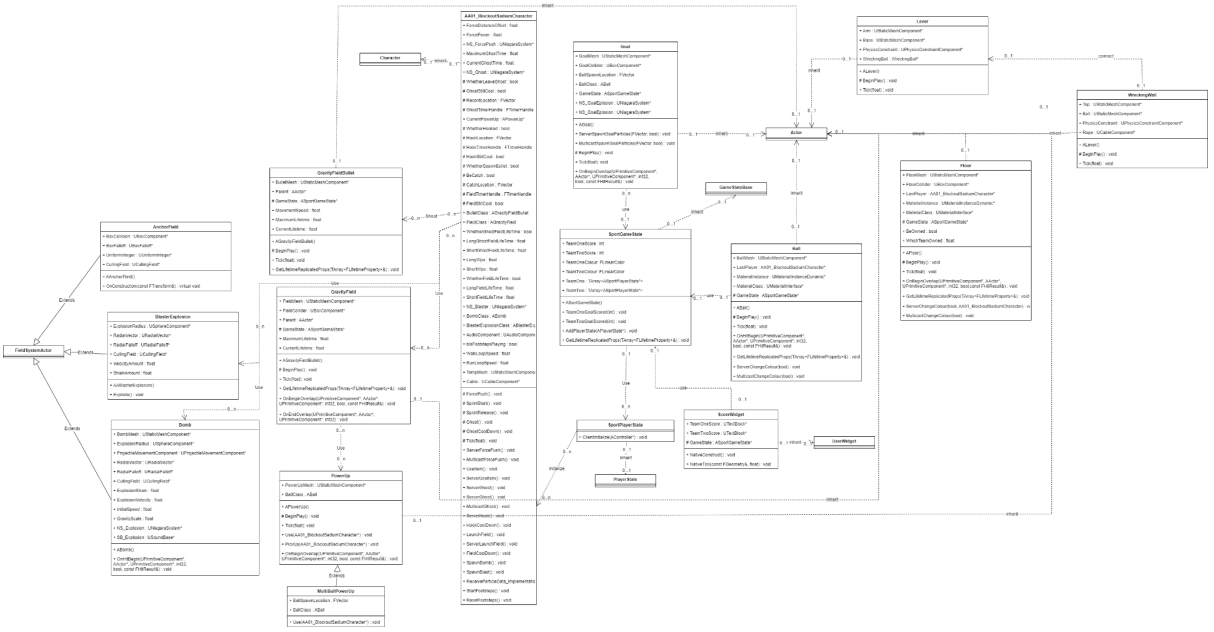
Game Mechanics

Gameplay Flowchart



Link: https://drive.google.com/file/d/1I52ijVSaiW9B-9GymuEsLgcaAkIaGBDG/view?usp=sharing

UML Diagram



Link:
https://drive.google.com/file/d/14EixmsISmkm4jqqJ1cRD64KvUG7-rwcx/view?usp=sharing

Movement Mechanics

Players can only move in the area enclosed by SM_Straight. The radius of the player is 42.f and the height is 96.0f. Players cannot overlap with other players, balls and goals. When players step on a floor block, the colour of the floor edges will become their team colour. When they move on the floor block that has their team colour, their speed will increase to 600(not applied in 1B).

- Walking

The player can press WASD on the keyboard to move forward, left, backward and right respectively. WASD is a very common key setting to control the movement of the character in the PC game. Also, this key setting can ensure that the player does not interfere with the use of the mouse when using the keyboard. The walk speed is 500. This speed can prevent the player from escaping the gravity field on foot too easily. The radius of the collision remains 42 and the height of the collision remains 96.

- Running

The player can hold Shift on the keyboard to run. When players are holding Shift, their movement speed will increase to 750. This value ensures that the player cannot run to the opposite side too quickly. Through this way, the player can focus more on using hooking or leaving the ghost to quickly move. When players release the key, their movement speed will go back to 500. When players are running, their collision will remain the same as the collision when they are walking and will follow their positions. When players are in the gravity field, they cannot run.

- Jumping

The player can press Space on the keyboard to jump. The jump velocity is set to 700. This speed can prevent the player from jumping outside of the edges. When players are jumping, the collision size will not change, but will follow their heights. When players are in the gravity field, they cannot jump.

Controls

Hook

- Key mapped: Q
  The player will use Hook very frequently during a match. Therefore, the key mapped is very close to the general movement, WASD.
- Axis binding: None
- Action binding: Current none
- Launch a rope and pull the player to the end of the rope
- Limits:
  There is 5 seconds of cooling to prevent the player from using Hook to move too quickly. There is a restriction of rope distance (1250) to avoid the player being able to pass the whole game world by Hook.

Ghost

- Key mapped: R
  Same reason as Hook.
- Axis binding: None
- Action binding: Current none
- Effects binding: NS_Ghost
- Leave a ghost at the current position. Press R again to go back to the ghost position.
- Limits:
  There is 5 seconds of cooling to prevent the player from using Ghost at random. The player only can go back to the ghost position in 5 seconds after he leaves a ghost.

Gravity Field

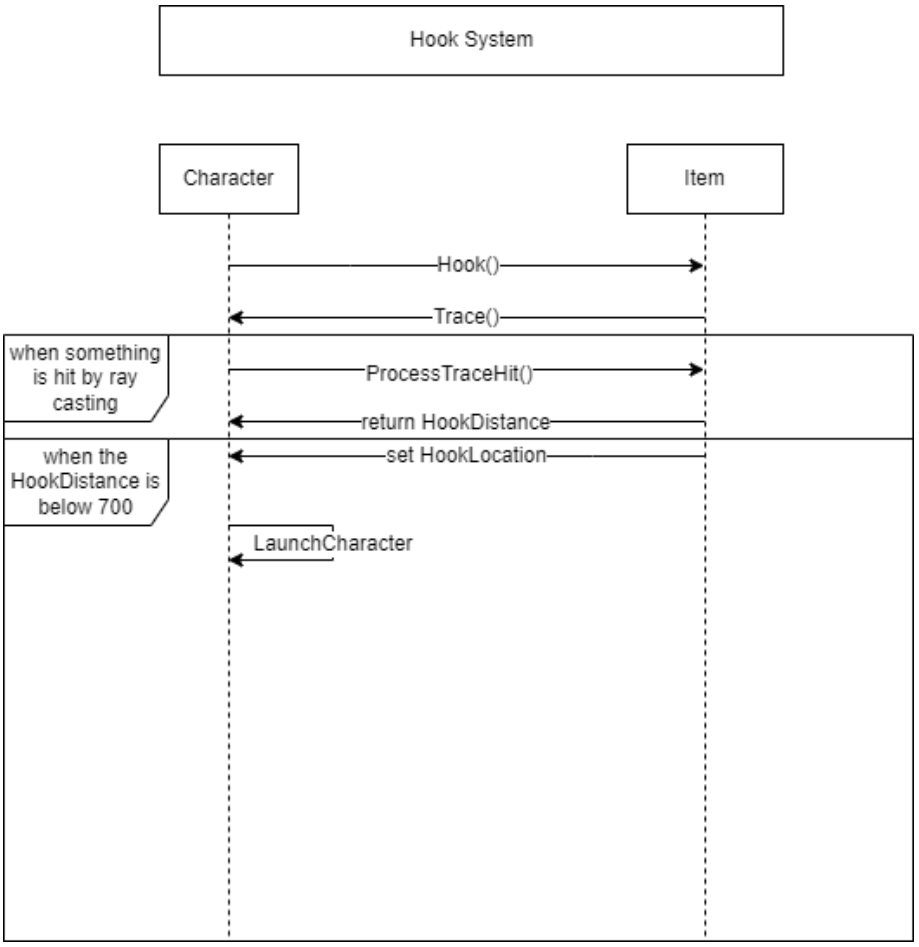
- Key mapped: Z
  The cooling time of launching a gravity field is 20 seconds. The frequency to use it is not very frequent in a match. Therefore, the key mapped is set to a relatively distant key.
- Axis binding: None
- Action binding: Current none
- Effects binding: NS_GravityField
- Launch a bullet towards the face direction. Press Z again to spawn the gravity field in the current position of the bullet.
- Limits:

There is 20 seconds of cooling to prevent the player from using Gravity Field at random. The gravity field bullet cannot change the trajectory of motion after it has been launched. Players only can spawn the gravity field in 5 seconds after they launch the bullet.

3 Core Gameplay Mechanics

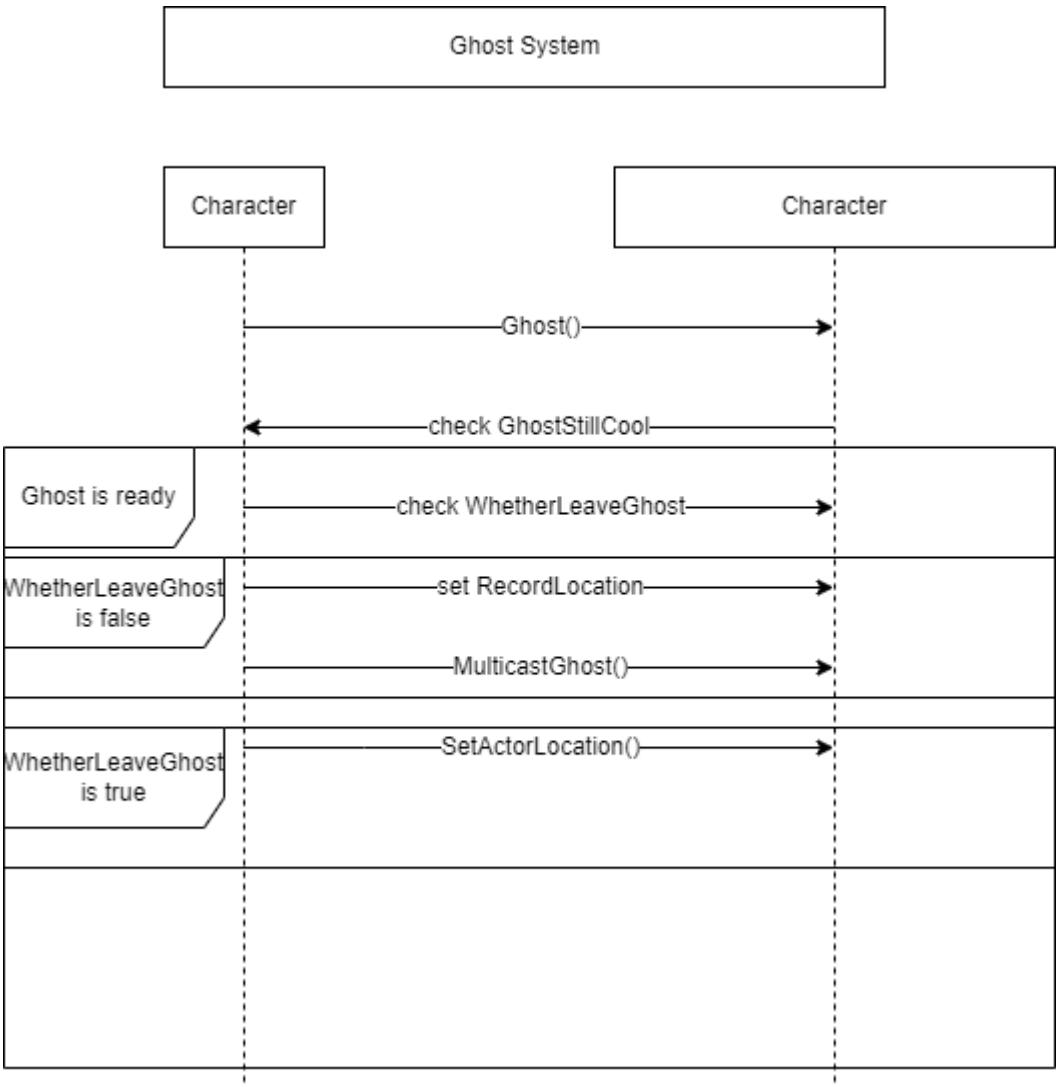
Hook

- The player can press Q on the keyboard to hook. When the player presses Q, it will check whether Hook is ready through HookStillCool. If it is ready, there will be a ray casting according to the mouse position on the screen. When the ray hits something, it will check whether the item hit is in the reasonable scope. If the distance between the item hit and the player is below 700, it will return true. The position of the item hit will be preserved as HookLocation. HookLocation - GetActorLocation will be preserved as VectorLength. Then, through LaunchCharacter(), the player will be pulled to HookLocation. Next, HOOK will go into a 5 second cooldown through SetTimer() which can be cleared by using Ghost. LaunchVelocity is calculated by VectorLength*DeltaTime*500.
- Through Hook, the player can move quickly to follow the ball and leave the ghost at a special position to prepare for returning.



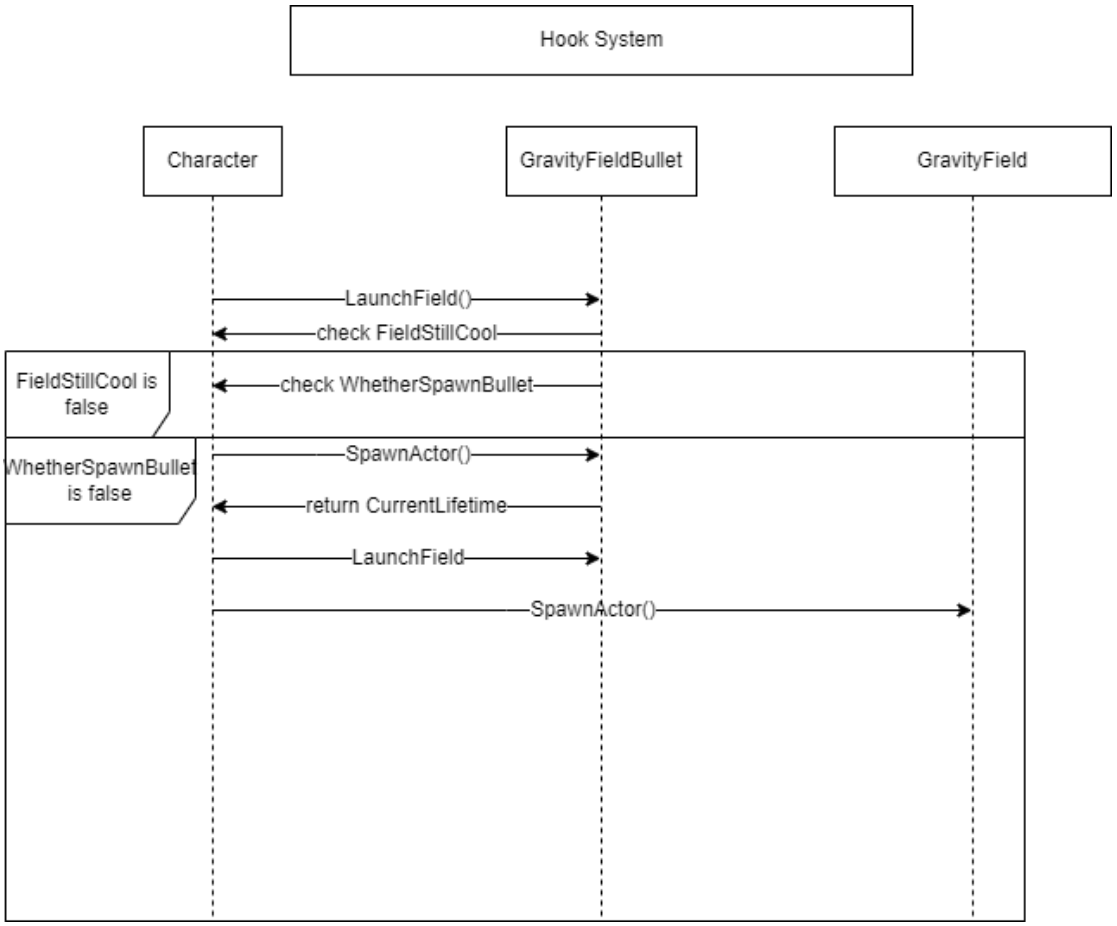
Ghost

- The player can press R on the keyboard to use Ghost. When the player presses R, it will check whether Ghost is ready through GhostStillCool. If it is true, it will check whether the player left a ghost 5 seconds ago through WhetherLeaveGhost. If it is false, the current position of the player will be preserved as RecordLocation. The Niagara System NS_Ghost will be spawned at RecordLocation. WhetherLeaveGhost becomes true. Then, the player has 5 seconds to press Q again. When the player presses Q again, the player can go back to RecordLocation. After the player goes back, there will be a ghost field to bounce off the rivals and the cooldown Hook will be cleared. Then Ghost will go into a 5 second cooldown through SetTimer().
- Through Ghost, the player is able to defend quickly and escape from the gravity field. The 5 seconds restriction also tests the player's release timing.



Gravity Field

- The player can press E on the keyboard to use Gravity Field. When the player presses E it will check whether Gravity is ready through FieldStillCool. If it is true, it will check whether the player shoots a bullet through WhetherSpawnBullet. If it is false, the player shoots a gravity field bullet. WhetherSpawnBullet becomes true. Then, the player can press E again. When he presses E again or the lifetime of the bullet finishes, there will be a gravity field generated at the position of the bullet. The lifetime of the gravity field is 15 seconds or 8 seconds, which depends on the duration of the bullet's flight. During its lifetime, it can pull the rival players towards its centre continuously. The lifetime of the gravity field will change according to the lifetime of the gravity field bullet. And Gravity Field will go into a 20 second cooldown through SetTimer().
- As the only skill which can restrict the rival players, the GravityField is a very powerful skill. Due to its relatively long cooling time, the player needs to decide when to use carefully and avoid the rivals escaping the field easily through Ghost.



Multiplayer

Replication of Game State and Player State

Gravity Field Bullet

```
void AGravityFieldBullet::GetLifetimeReplicatedProps(TArray<FLifetimeProperty>& OutLifetimeProps) const
{
    Super::GetLifetimeReplicatedProps([&] OutLifetimeProps);
    DOREPLIFETIME(AGravityFieldBullet, Parent);
}
```

Through GetLifetimeReplicateProps(), Unreal knows that there are Replicated components inside of the Gravity Field Bullet class, and which variables to replicate.

```
ASportGameState* GameState;
```

```
GameState= Cast<ASportGameState>( Src: GetWorld()->GetGameState());
```

Through getting the game state from getWorld(), the Gravity Field Bullet class can be at the same game state as other actors.

```
bReplicates = true;
BulletMesh ->SetIsReplicated( ShouldReplicate: true);
NetPriority=3.0f;
```

In the Gravity Field Bullet class, BulletMesh is set as Replicated. It can ensure that the client and the server can display the bullet mesh as well as its position simultaneously and correctly.

```
UPROPERTY(Replicated,EditAnywhere)AActor* Parent;
```

Parent is being replicated. It can ensure that when the game world has bullets from different players, the correct bullet can be detonated by the correct corresponding player in both server and client.

Gravity Field

```
void AGravityField::GetLifetimeReplicatedProps(TArray<FLifetimeProperty>& OutLifetimeProps) const
{
    Super::GetLifetimeReplicatedProps( [&] OutLifetimeProps);
    DOREPLIFETIME(AGravityField, Parent);
}
```

The same as the Gravity Field Bullet class.Through GetLifetimeReplicateProps(), Unreal knows that there are Replicated components inside of the Gravity Field class, and which variables to replicate.

```
ASportGameState* GameState;
```

```
    GameState= Cast<ASportGameState>( Src: GetWorld()->GetGameState());
    FieldCollider->OnComponentBeginOverlap.AddDynamic(this,&AGravityField::OnBeginOverlap);
    FieldCollider->OnComponentEndOverlap.AddDynamic(this, &AGravityField::OnEndOverlap);
```

Through getting the game state from getWorld(), the Gravity Field class can be at the same game state as other actors, such as Gravity Field Bullet.

```
bReplicates = true;
FieldMesh ->SetIsReplicated( ShouldReplicate: true);
FieldCollider->SetIsReplicated( ShouldReplicate: true);
NetPriority=3.0f;
```

```
{
    if(HasAuthority())
    {
        if(OtherActor && OtherActor != thi
```

In the Gravity Field class, FieldMesh is set as Replicated. It can ensure that the client and the server can display the field mesh correctly.
Also, FieldCollider is set as Replicated. It can ensure that the size of overlap can work correctly in both client and server. By HasAuthority(), the correct actor which has network authority can be impacted by overlap.

```
UPROPERTY(Replicated,EditAnywhere)AActor* Parent;
```

Parent is being replicated. It can ensure that the gravity field cannot impact the player who launches this field in both server and client.

Security / Anti-Cheat Identification

Through my game experience, some players will hide their skill display effects. Therefore, I need to ensure all skill effects can be displayed in the correct positions and maintain the correct lifetime, which means focusing more on the mesh and the Niagara System displayed in the server and the client.
Also, some players will use skills with no cooldown. Hence, I need to ensure that all skill cooldowns can be applied correctly in both server and client, which means FTimerHandle needs to be solved seriously.

Security / Anti-Cheat Prevention

```
UFUNCTION(Server, Reliable)void ServerGhost();
UFUNCTION(NetMulticast, Unreliable)void MulticastGhost();
```

Through the keyword NetMulticast, the Niagara effect of ghost can be displayed correctly in both server and client, which avoids the incorrect show of Ghost effects.

```
FieldCollider->SetupAttachment(FieldMesh);
bReplicates = true;
FieldMesh ->SetIsReplicated( ShouldReplicate: true);
FieldCollider->SetIsReplicated( ShouldReplicate: true);
NetPriority=3.0f;
```

Through bReplicates and SetIsReplicated(true), the mesh of the field and its collision size can be replicated in the server. This can prevent the display scope and the actual effect scope are not the same in different clients.

```
FTimerHandle FieldTimerHandle;
```

About the cooldowns, I will add the keyword Replicated to ensure that cooldowns can be controlled or be synchronised by the server. This can avoid cooldown time if each player is not the same in different clients.

Shader Effects

M_Floor

- Overview of Effect

M_Floor is a material for each floor block in the match. Its purpose is to help players distinguish quickly which floor block is occupied by their own team or the rival team. It can help players plan their paths to move more quickly during the match and increase the walk speed of the player.

- Effect Description

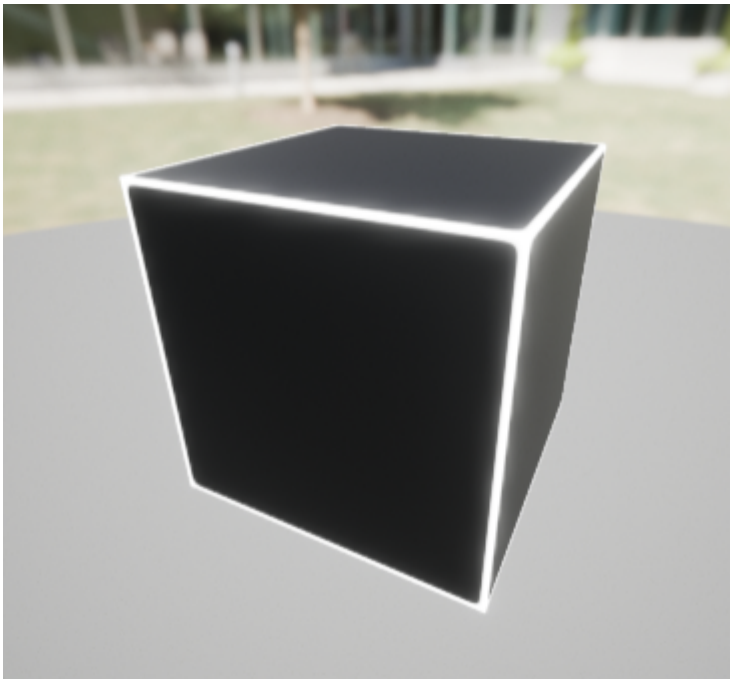
Description of what the effect looks like

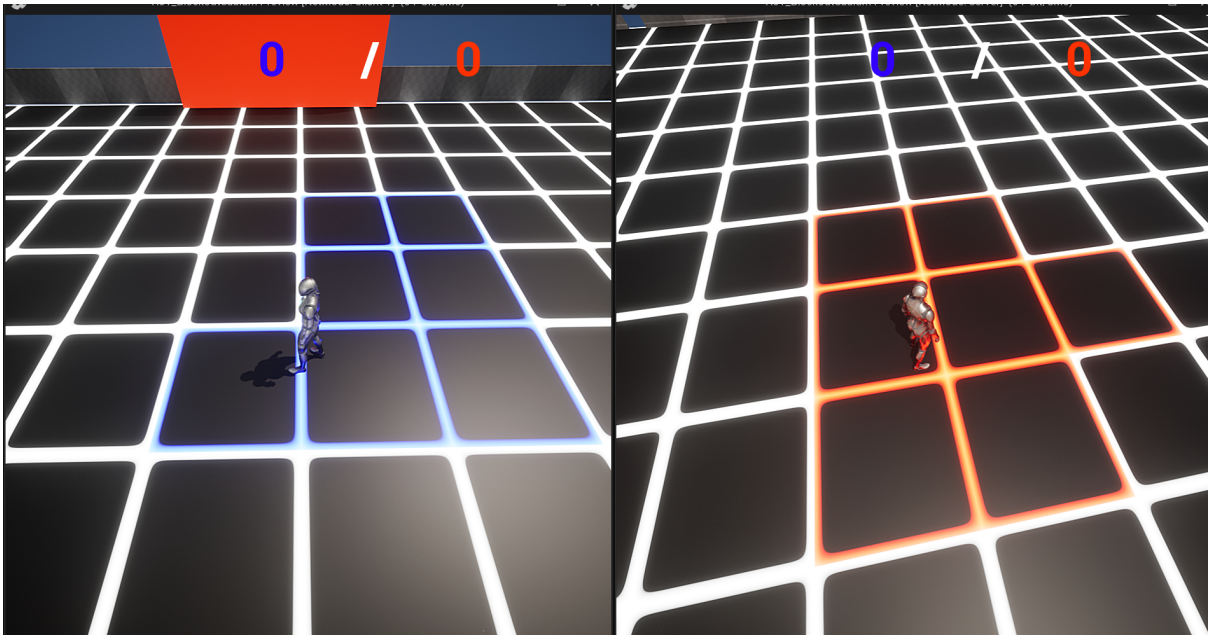
The colour of the main body of the floor is black. The edges can glow during the match. The initial emissive colour is white. Then the emissive colour will change to blue or red according to which team steps on this floor.

Reference images used in the creation of the effect

Reference Link:
https://www.computerfloorpros.com/raised-floor-systems/raised-floor-applications/server-room/

Screenshots of effect in-game



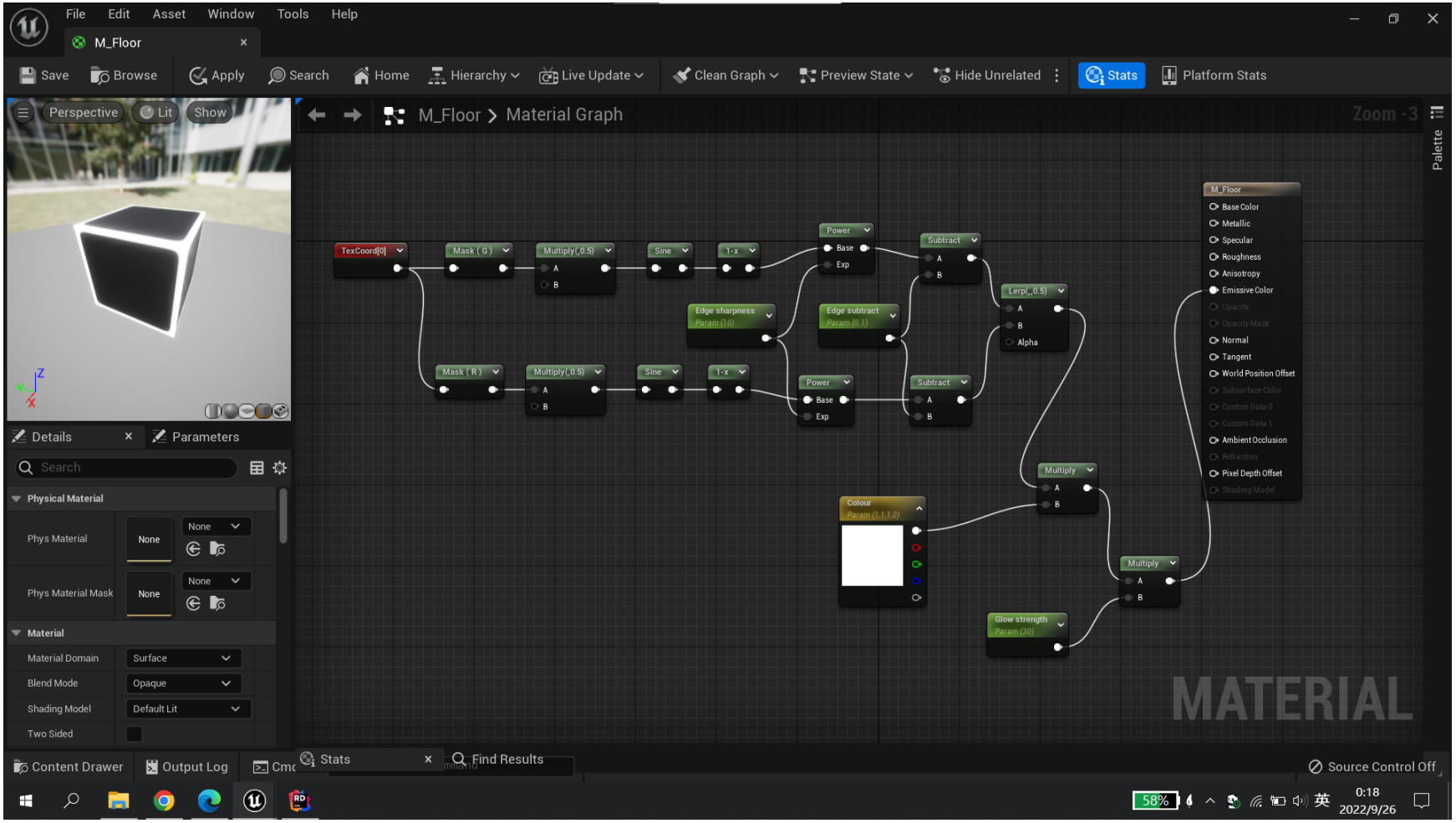


The properties used & Concrete values

The material domain set is Surface. The blend mode set is Opaque. The shading model is Default Lit. The other properties are left as default. The initial colour of the edges is white. The edge sharpness is 10, the edge subtract is 0.1 and the glow strength is 30.

- Node Graph / HLSL Breakdown

Screenshots of Node Graph



Explanation of the process & Annotations for each of the nodes used

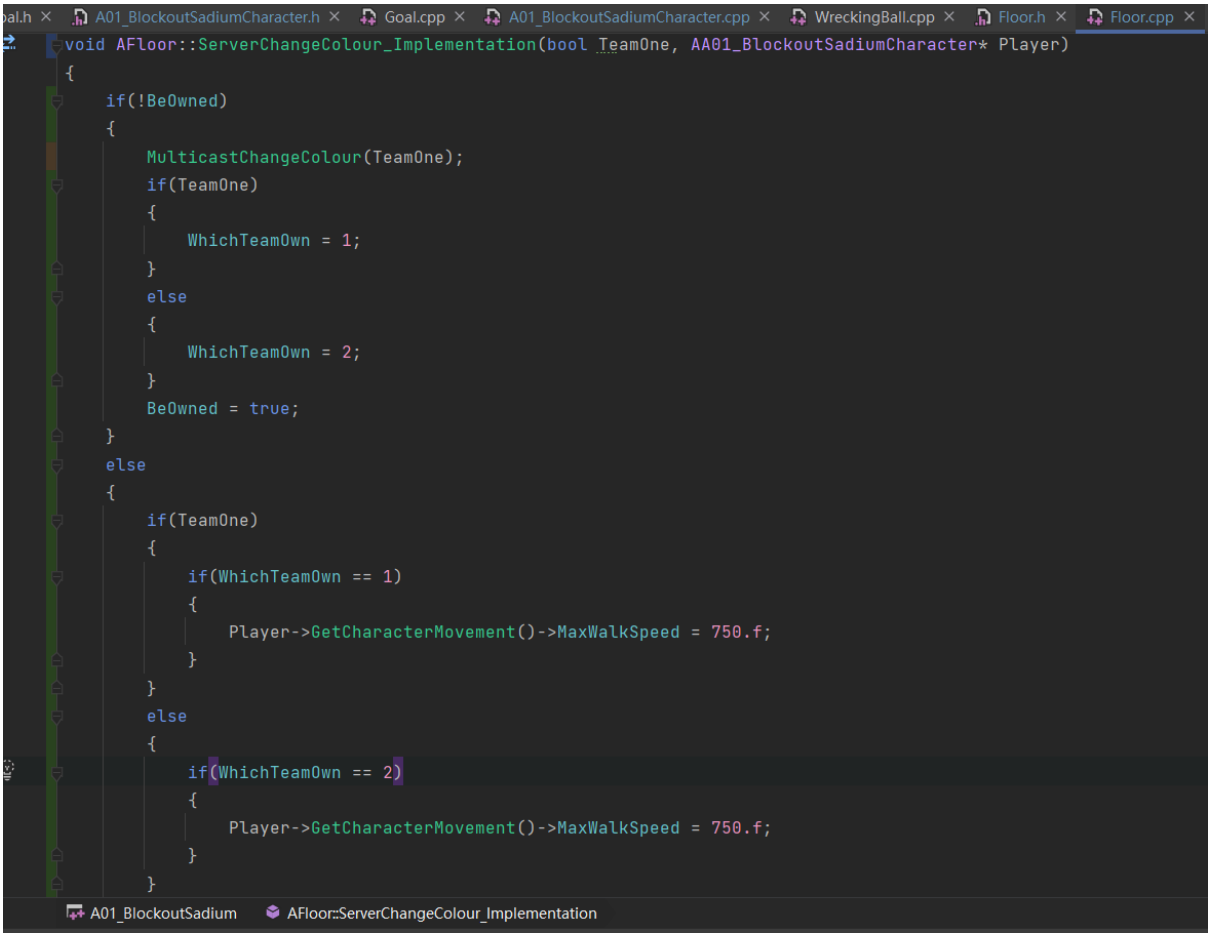
The path of Mask(G) will act on the horizontal edges of the cube. And the path of Mask(R) will act on the vertical edges of the cube.

1. The value of the edge sharpness can impact the thickness of the edges. Higher value will result in smaller thickness.
2. The value of the edge subtract can change the impact on the corners. 0.1 is a suitable value to cover all the edges.
3. The value of the glow strength can impact the light intensity. Higher value will result in higher intensity.

Details of custom material functions used in the effect

The custom material I used is Colour(Para4). The initial value is (1, 1, 1, 0). During the match, Colour will change through OnBeginOverlap(). Its value will use FLinearColor::Blue and FLinearColor::Red. These two values are the same colour as the change of the ball, which ensures that the team colour can keep consistent on the floor and ball during the match. When the player walks on the floor which has the same team colour, their walk speed will be 750.

```
void AFloor::OnBeginOverlap(UPrimitiveComponent* OverlappedComponent, AActor* OtherActor,
    UPrimitiveComponent* OtherComponent, int32 OtherBodyIndex, bool bFromSweep, const FHitResult& SweepResult)
{
    if(OtherActor && OtherActor != this)
    {
        if(Cast<AA01_BlockoutSadiumCharacter>(OtherActor))
        {
            if(HasAuthority())
            {
                LastPlayer = Cast<AA01_BlockoutSadiumCharacter>(OtherActor);
                if(GameState)
                {
                    if(GameState->TeamOne.Contains( Item: LastPlayer->GetPlayerState()))
                    {
                        ServerChangeColour( TeamOne: true);
                    }
                    else
                    {
                        ServerChangeColour( TeamOne: false);
                    }
                }
            }
        }
    }
}
```


oal.h
A01_BlockoutSadiumCharacter.h
Goal.cpp
A01_BlockoutSadiumCharacter.cpp
WreckingBall.cpp
Floor.h
Floor.cpp
void AFloor::ServerChangeColour_Implementation(bool TeamOne, AA01_BlockoutSadiumCharacter* Player)
{
if(!BeOwned)
{
MulticastChangeColour(TeamOne);
if(TeamOne)
{
WhichTeamOwn = 1;
}
else
{
WhichTeamOwn = 2;
}
BeOwned = true;
}
else
{
if(TeamOne)
{
if(WhichTeamOwn == 1)
{
Player->GetCharacterMovement()->MaxWalkSpeed = 750.f;
}
}
else
{
if(WhichTeamOwn == 2)
{
Player->GetCharacterMovement()->MaxWalkSpeed = 750.f;
}
}
A01_BlockoutSadium
AFloor::ServerChangeColour_Implementation

M_Bullet

● Overview of Effect

M_Bullet is a material for the gravity field bullet in the match. Its purpose is to make the bullet more conspicuous and help players distinguish quickly which bullet belongs to their own team or the rival team. It can help players focus more on the motion of the bullet and avoid the formation of the gravity field in advance.

● Effect Description

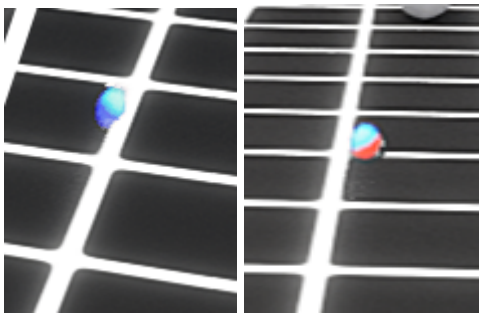
Description of what the effect looks like

The main tone of M_Bullet is cartoon material. It looks like a capsule toy containing something, which conforms to the spawn of the gravity field. The base colour will change to blue or red according to the team.

Reference images used in the creation of the effect

Reference Link:
https://www.amazon.com/SmileMakers-Empty-Capsule-Mix-Prizes-Giveaways-250/dp/B07D812S9H

Screenshots of effect in-game

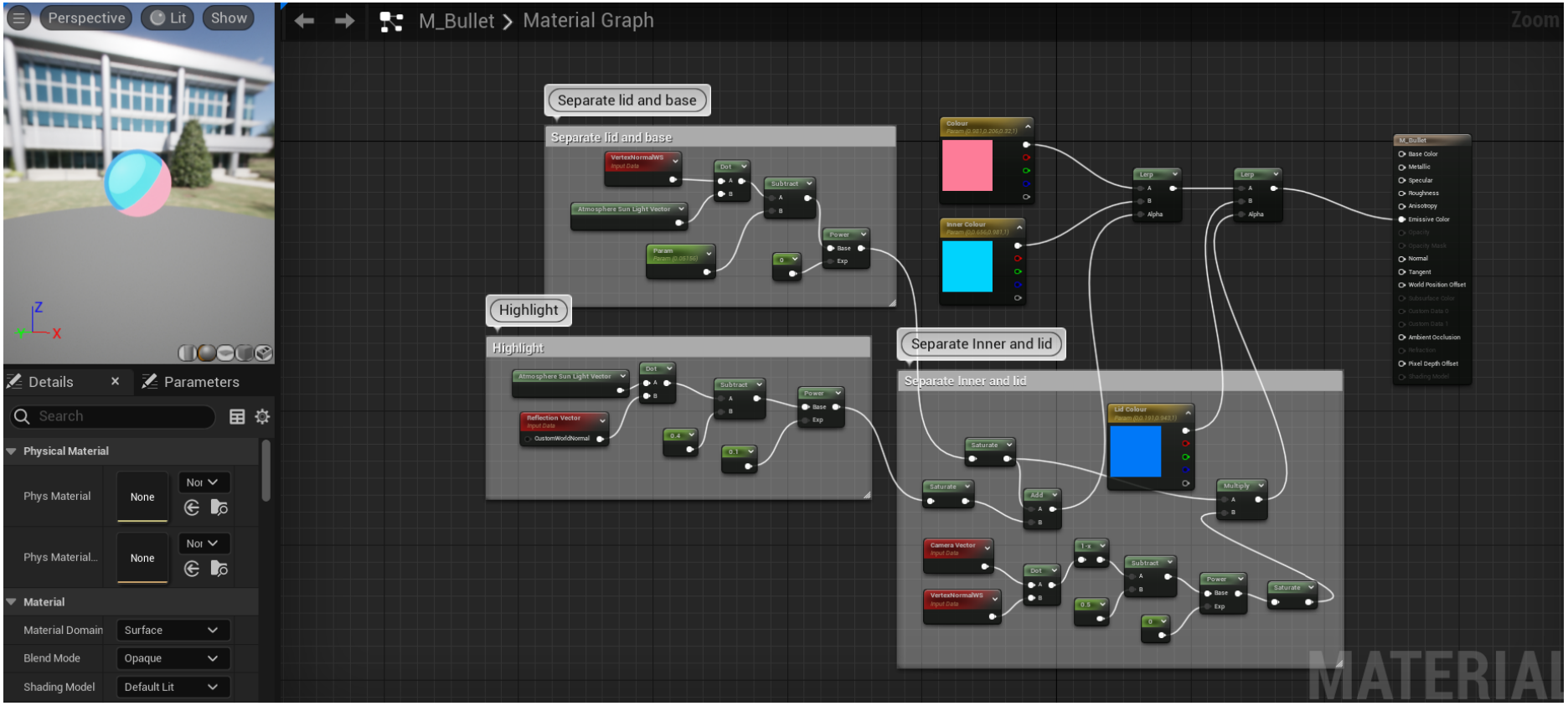


The properties used & Concrete values

The material domain set is Surface. The blend mode set is Opaque. The shading model is Default Lit. The other properties are left as default. The initial base colour is set as Param(0.981, 0.206, 0.32, 1). The lid colour is set as Param(0, 0.191, 0.943, 1). The inner colour is set as Param(0, 0.656, 0.981, 1).

- Node Graph / HLSL Breakdown

Screenshots of Node Graph



Explanation of the process & Annotations for each of the nodes used

The section named “Separate lid and base” can impact the volume allocated of the lid and base on this sphere. The section named “Highlight” can impact the volume of the inner reflection. The section named “Separate Inner and lid” can impact the volume of the inner in the lid.

Colour will be the colour of the base. Lid Colour will be the colour of the lid. Inner Colour of the inner.

Details of custom material functions used in the effect

The custom material I used is Colour(Para4). The initial value is (0.981, 0.206, 0.32, 1). Through MultiCastChangColour(), Colour will change to FLinearColor::Blue and FLinearColor::Red which are stored in GameState according to the team colour. These two values are the same colour as the change of the floor and the ball, which ensures that the team colour can keep consistent during the match.

```
    }
    if(HasAuthority())
    {
        if(GameState)
        {
            if(GameState->TeamOne.Contains(Item: Cast<AA01_BlockoutSadiumCharacter>(Parent)->GetPlayerState()))
            {
                ServerChangeColour(TeamOne: true);
            }
            else
            {
                ServerChangeColour(TeamOne: false);
            }
        }
    }
}
```

```
void AGravityFieldBullet::MulticastChangeColour_Implementation(bool TeamOne)
{
    if(MaterialInstance)
    {
        if(TeamOne)
        {
            MaterialInstance->SetVectorParameterValue("Colour", GameState->TeamOneColour);
        }
        else
        {
            MaterialInstance->SetVectorParameterValue
            ("Colour", GameState->TeamTwoColour);
        }
    }
}
```

Post Processing Effects

GlobalPostProcess

- Overview of Effect

Just like the name of this effect, this effect is a global post processing effect to impact the sharpness and blurring of the whole level. Its purpose is to increase the experience and look of the game screen. The increase in the sharpness can highlight the display of bullets and the ball to notice players during the game.

- Effect Description

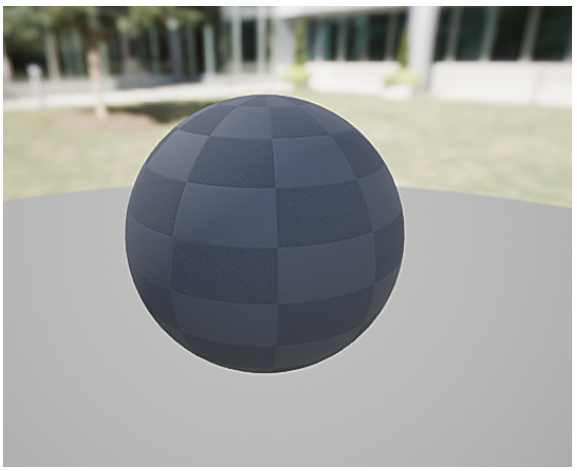
Description of what the effect looks like

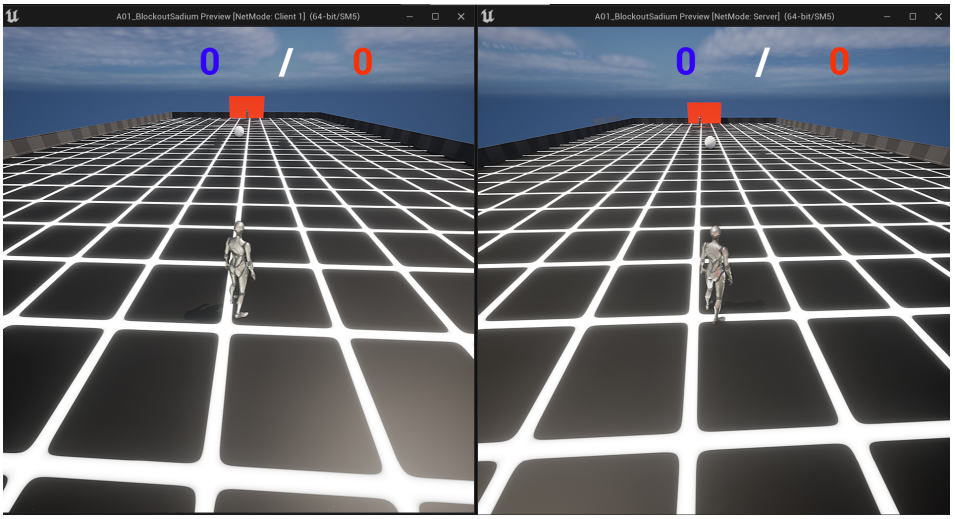
The clarity of the whole level will increase and all objects in the level will sharpen up. The player display screen will become clear. There will be an outline of each object to highlight the object.

Reference images used in the creation of the effect

Reference Link:
https://www.dbltap.com/posts/overwatch-fan-creates-overball-game-mode-in-workshop-01djxb6an6r4

Screenshots of effect in-game



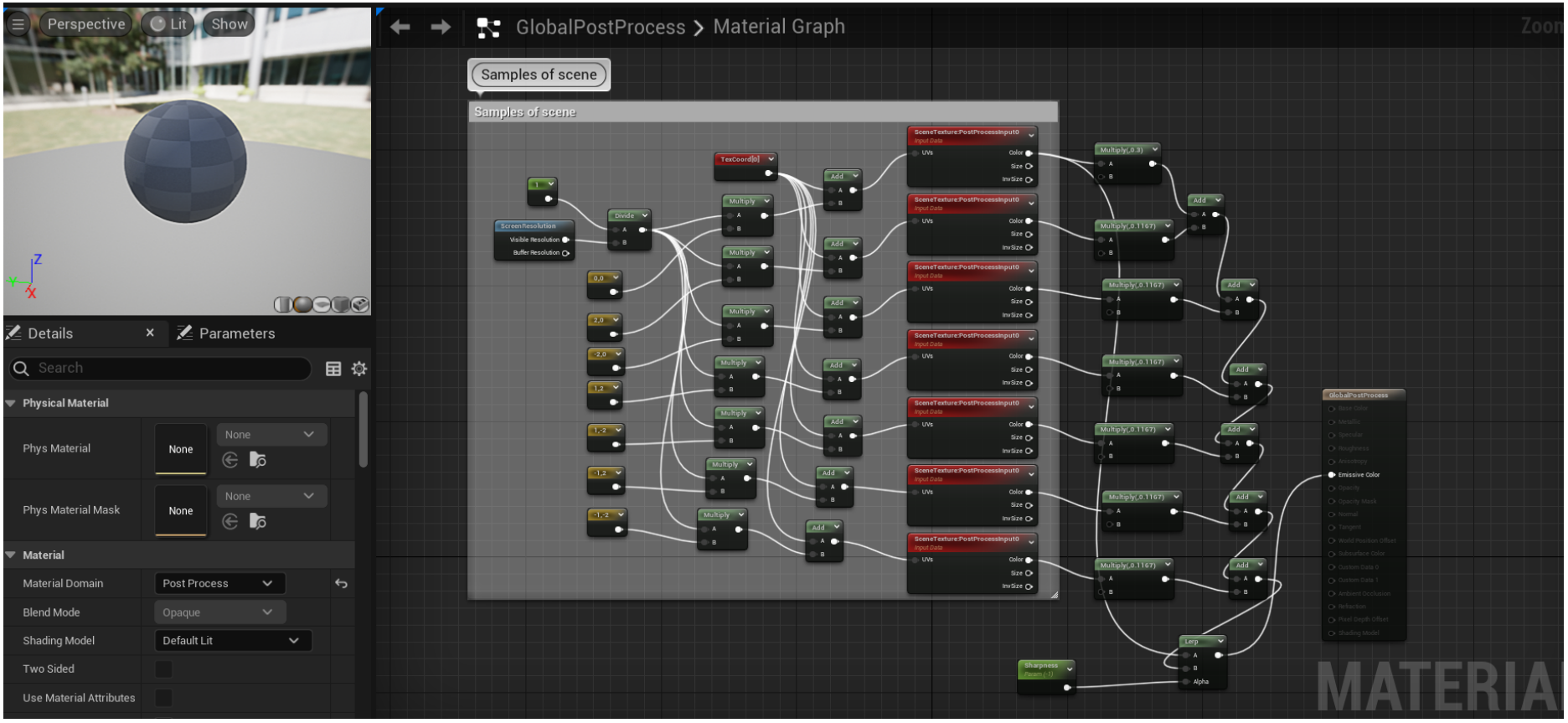


The properties used & Concrete values

The post process material used is GlobalPostProcess. Infinite Extent(unbound) is ticked to ensure that the effect applies to the entire scene. The material domain is set to Post Process. The shading Model is Default Lit. The other properties are left as default. Sharpness is set to -1.

- Node Graph / HLSL Breakdown

Screenshots of Node Graph



Explanation of the process & Annotations for each of the nodes used

The section named “Samples of scene” will get seven sets of uv coordinates to take samples of the scene in the centre and in six additional positions around the centre. Then through Multiply and Add, I combine those 7 samples together and weight the centre sample (0.3) slightly more than each of the samples (0.1167) around it. I use Lerp and a constant to control the effect. The constant is named as Sharpness. The constant is set to -1, which makes the scene sharper. When it is zero, there will be no effect and when it is positive, the scene will be fuzzy.

Details of custom material functions used in the effect

The custom material can be Sharpness. In the later development, the value of Sharpness can change during the game to help players experience the screen quality which is more suitable to them.

LocalPostProcess

- Overview of Effect

LocalPostProcess is located at the centre of the game map. It can distort the player's view into this area. The purpose is to increase the difficulty of dribbling or intercept when passing the centre. For gameplay, I use it to increase the intensity of confrontation between players when they are in the centre and accelerate the game process.

- Effect Description

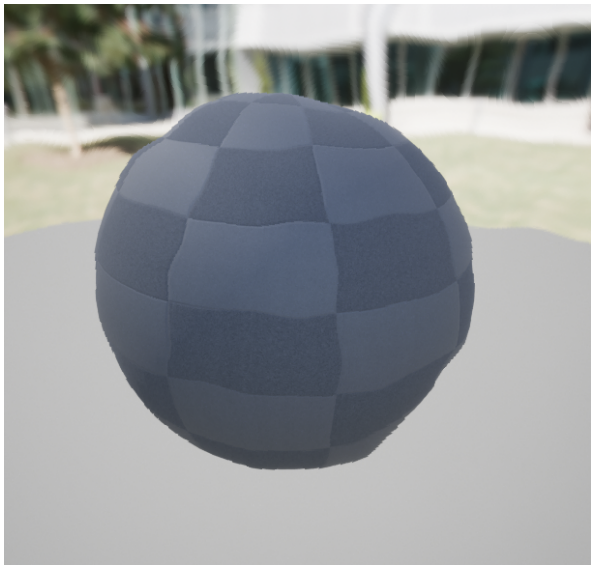
Description of what the effect looks like

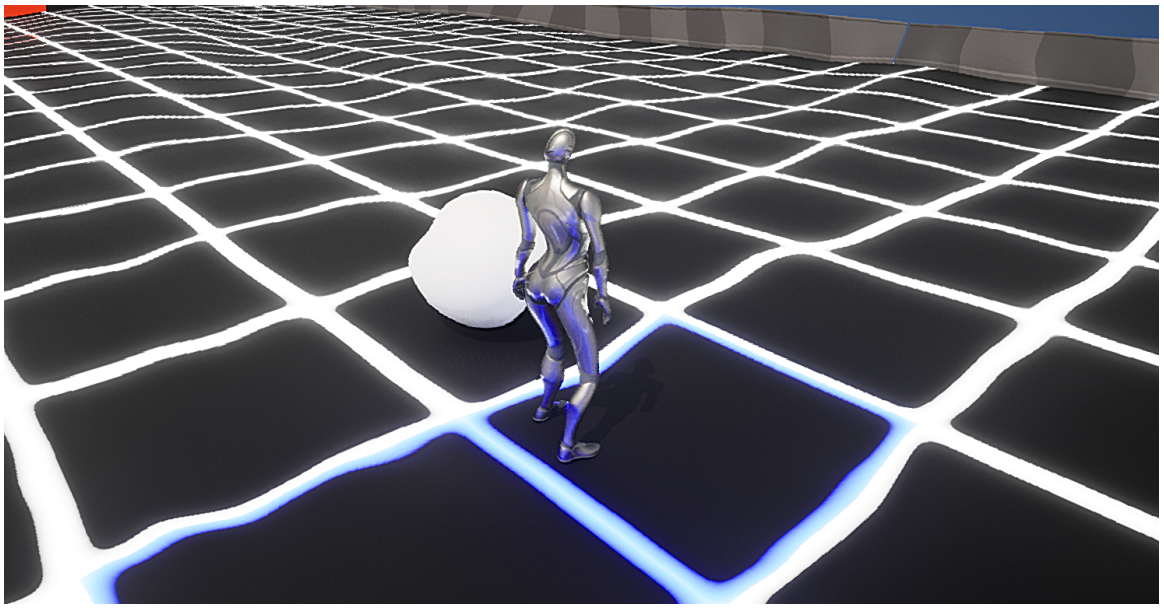
When players enter the centre of the game map, their screen will begin to distort continually and other elements, such as colour tone, will not change.

Reference images used in the creation of the effect

Reference Link:
https://www.stevenaitchison.co.uk/7-thinking-patterns-use-distort-reality/

Screenshots of effect in-game



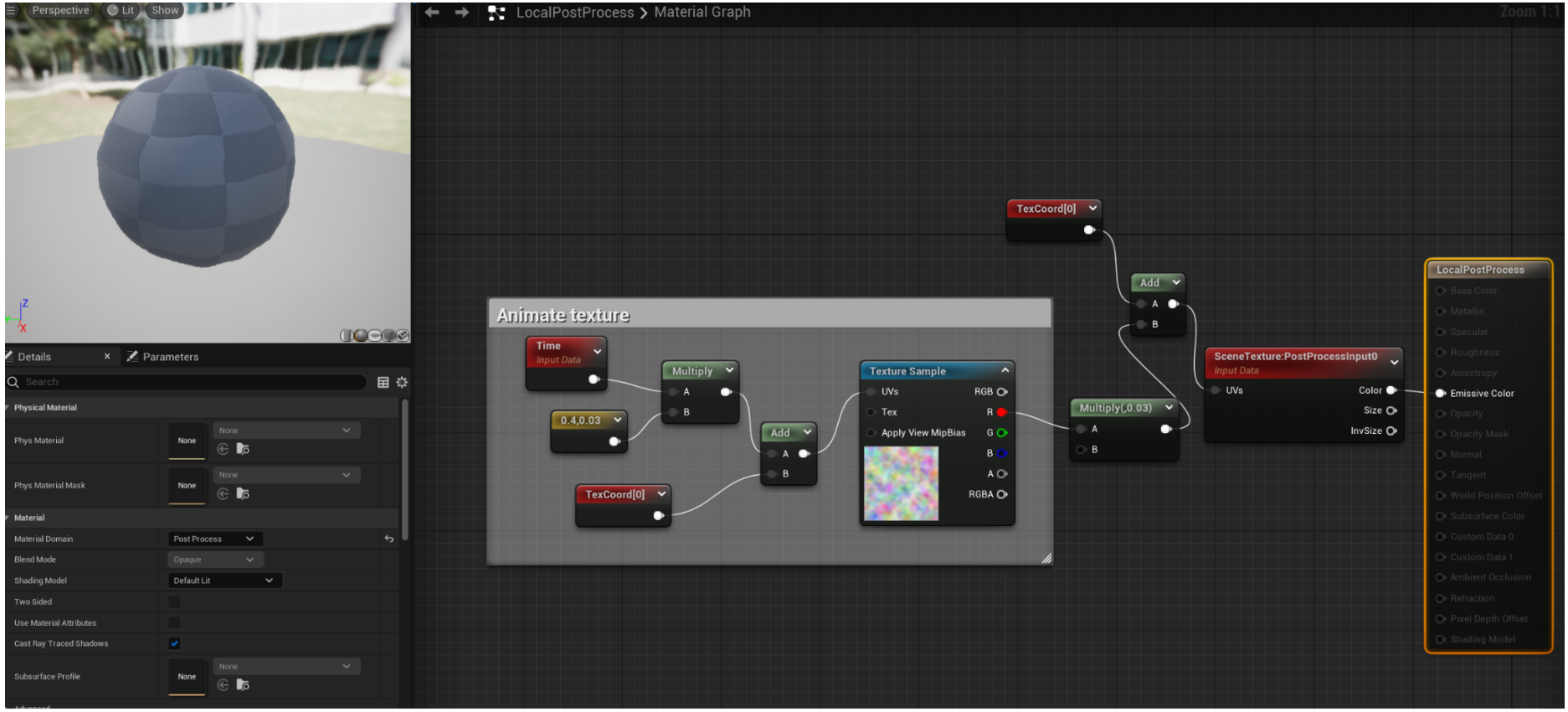


The properties used & Concrete values

The post process material used is LocalPostProcess. The location is set to (1440, 3710, 90). The scale is set to (17.75, 4.75, 12.75). Those two values can ensure the local post process volume can cover the centre of the game map. The material domain is set to Post Process. The shading Model is Default Lit. The other properties are left as default. The texture sample I used is game_win_noise.

- Node Graph / HLSL Breakdown

Screenshots of Node Graph



Explanation of the process & Annotations for each of the nodes used

In the section named “Animate Texture”, I sample a texture, game_win_noise and use Time to set the animation of the texture. Then I multiply the result by 0.03 and turn it down a bit. Through Add, I combine the result with texture coordinates.

Details of custom material functions used in the effect

The custom material I used is the constant vector 2. Its value can impact the distorting speed of the effect. And in the later development, its value will increase along with the match time.

Particle Effects

NS_GhostField

- Overview of Effect

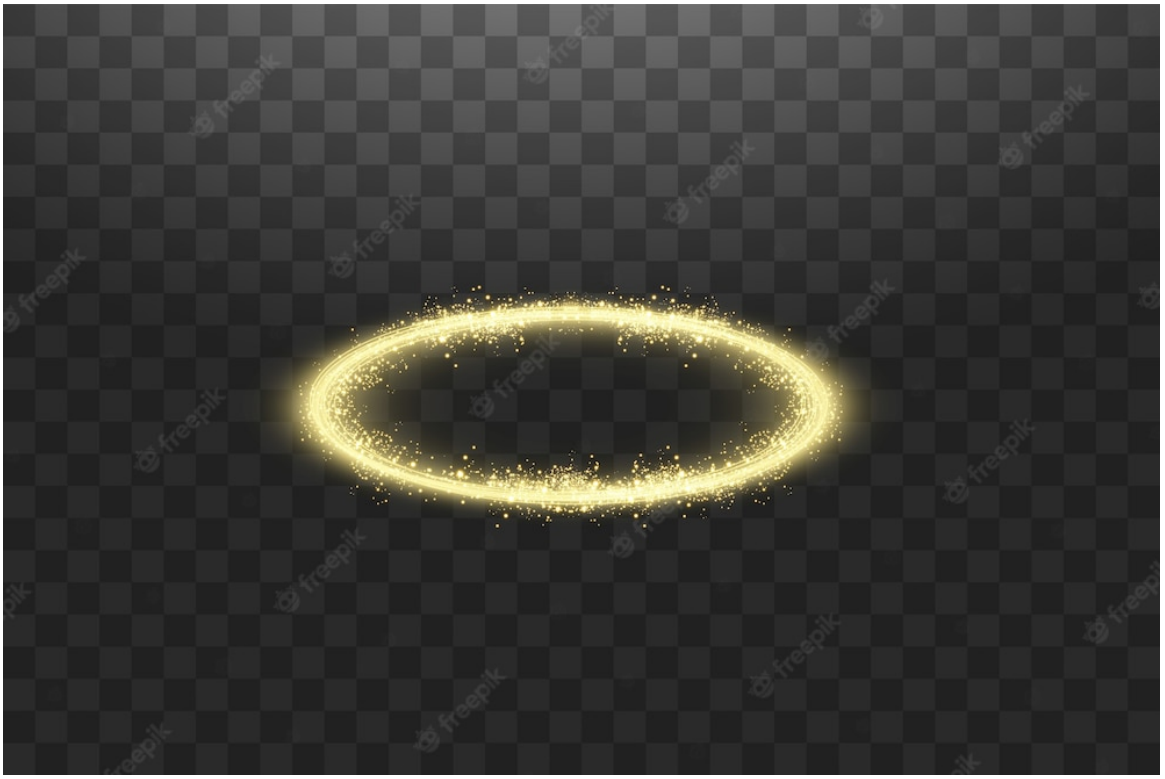
NS_GhostField is the particle effect to display the ghost field when the player goes back to the ghost field. Its purpose is to make the rival players notice the effective area of Ghost.

- Effect Description

Description of what the effect looks like

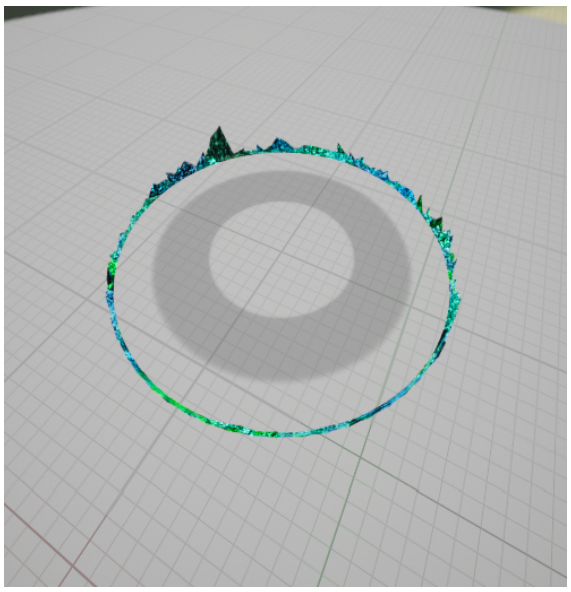
NS_GhostField looks like a halo that spreads outward and upward. And the halo is made up of glitter chips.

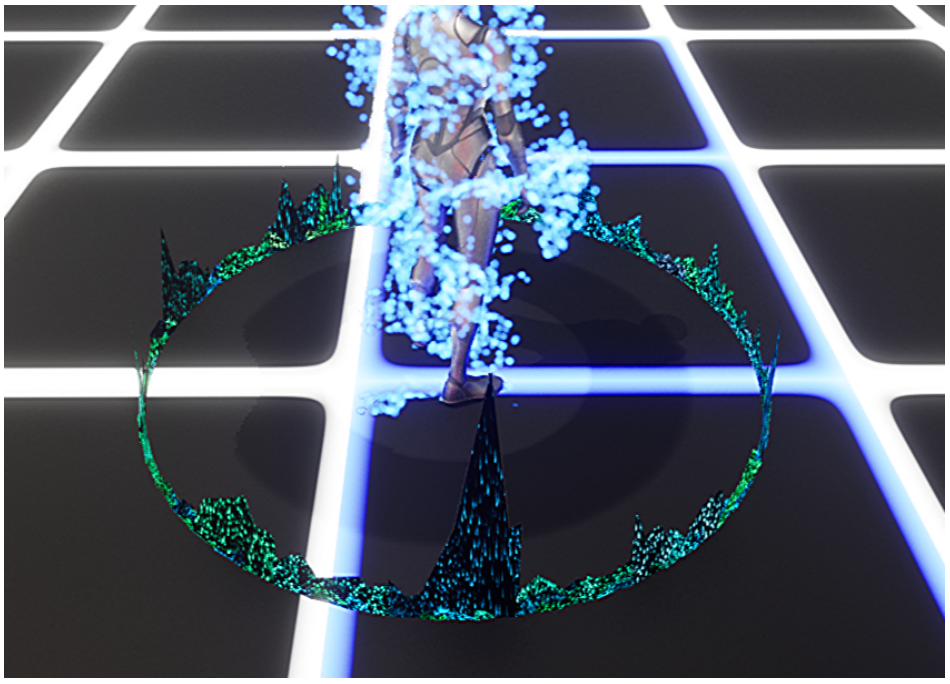
Reference images used in the creation of the effect



Reference Link:
https://www.freepik.com/premium-vector/golden-halo-angel-ring-isolated-dark_10930897.htm

Screenshots of effect in-game



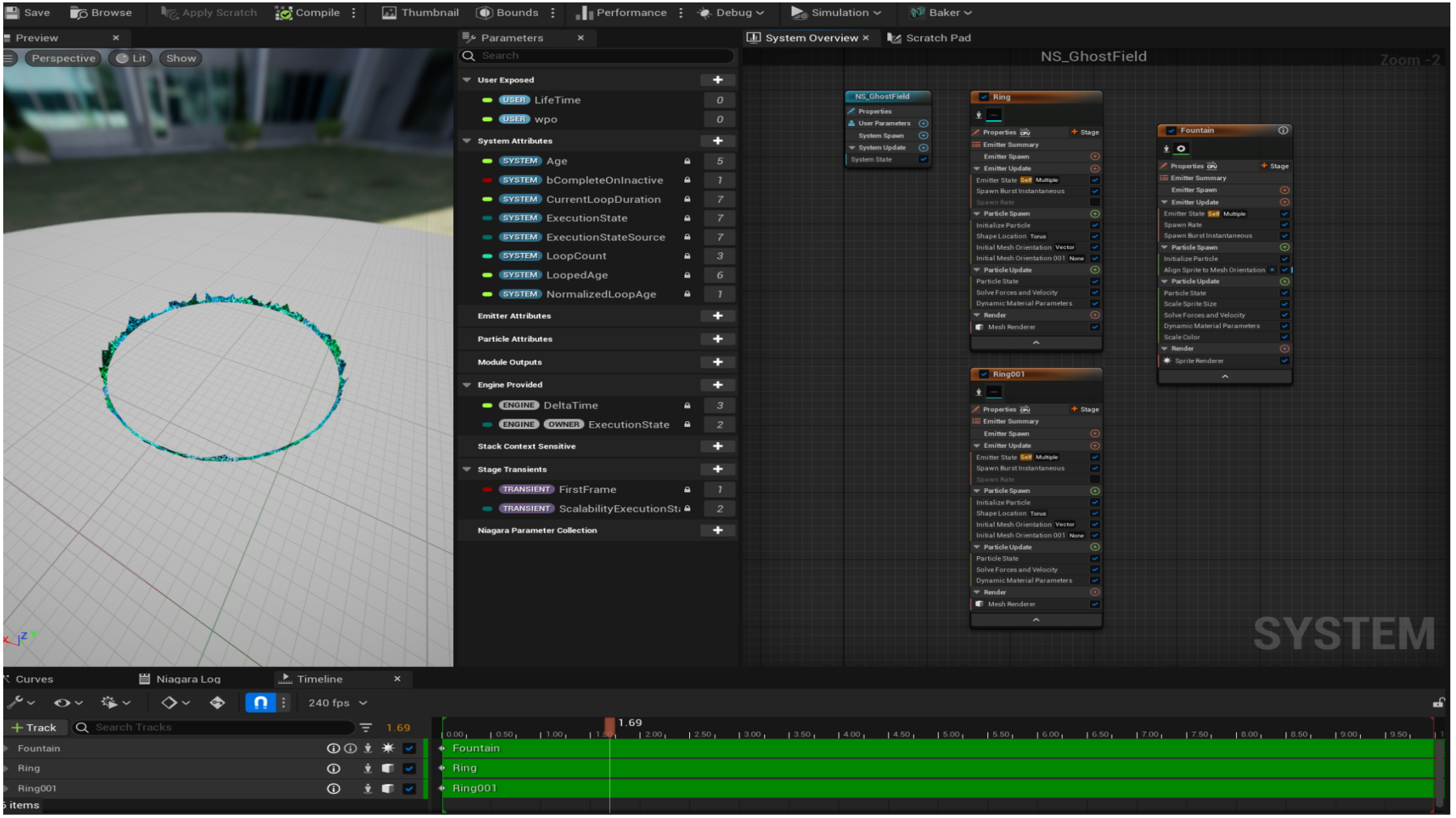


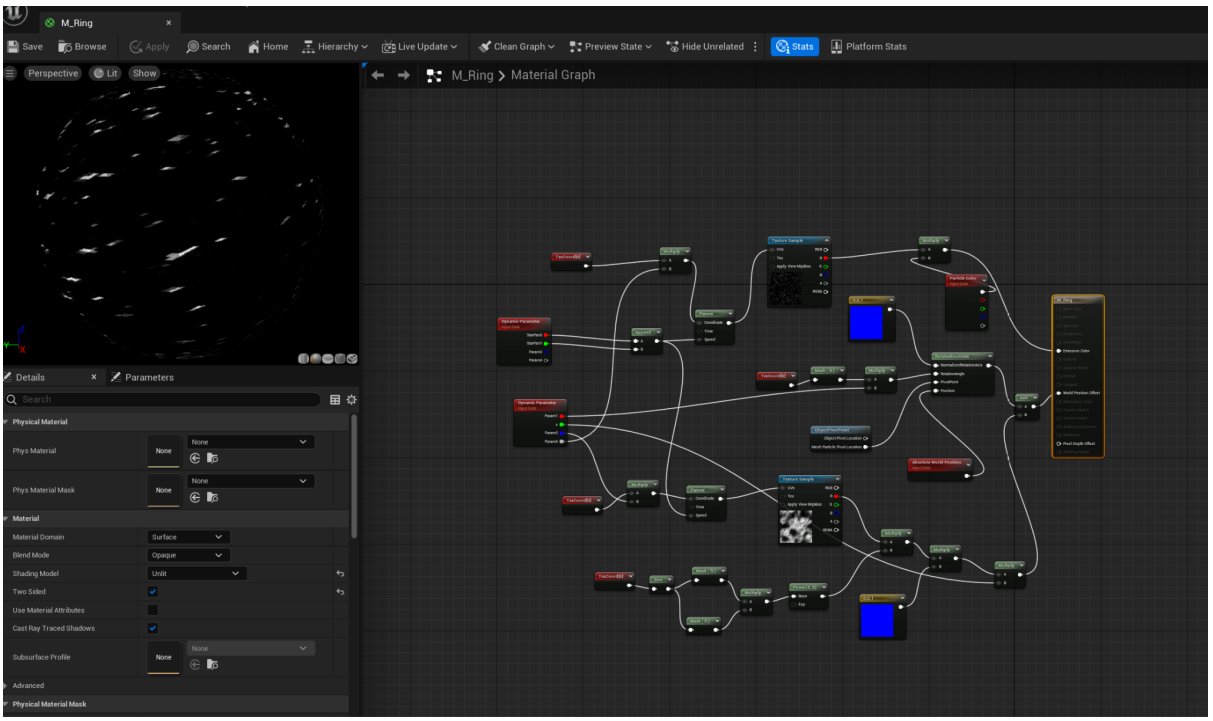
The properties used & Concrete values

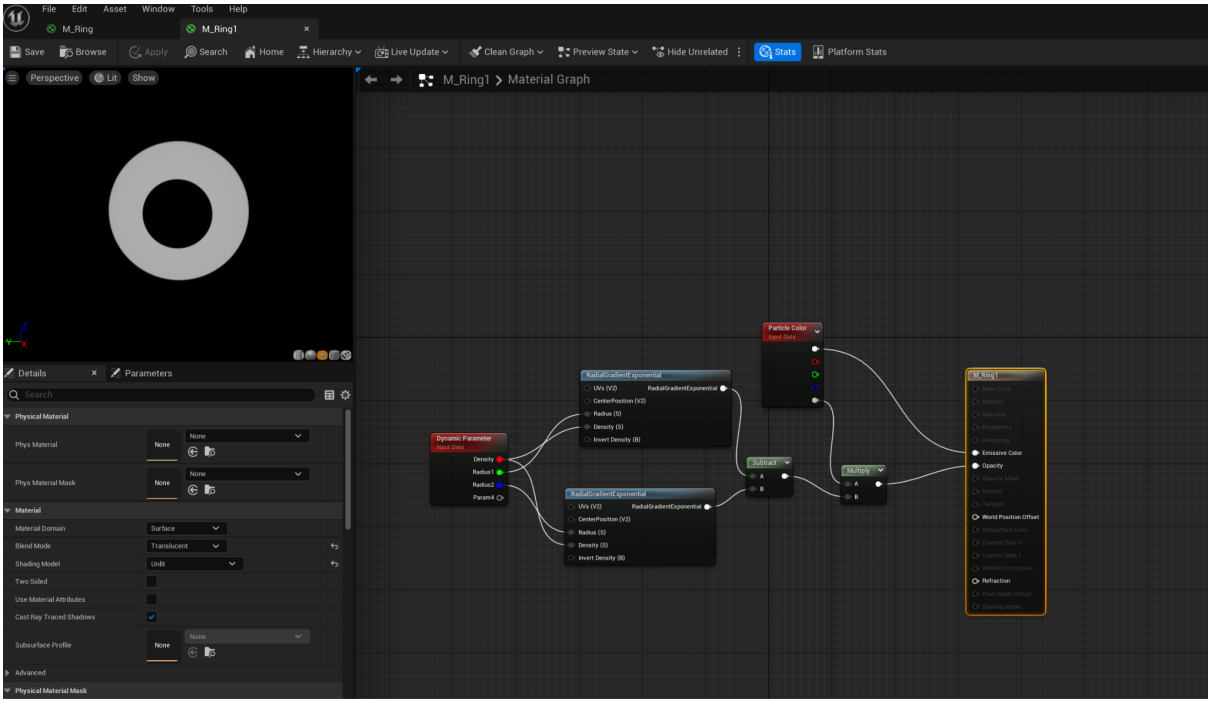
Max Pool Size is set to 32. When emitters finish, the Niagara system will be killed. The other properties are left as default. There are three emitters. The user exposed parameters are LifeTime and wpo. They impact the lifetime and the display intensity of the Niagara system respectively. The materials I used are M_Ring and M_Ring1.

- Niagara System / Emitters Breakdown

Screenshots of Niagara System and Emitter

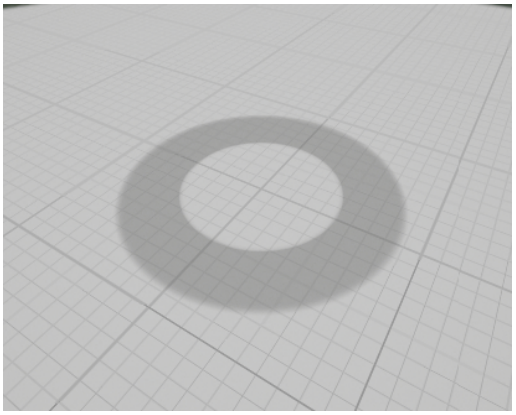



M_Ring
Save
Browse
Apply
Search
Home
Hierarchy
Live Update
Clean Graph
Preview State
Hide Unrelated
Stats
Platform Stats
Perspective
Lit
Show
M_Ring > Material Graph
Details
Parameters
Search
Physical Material
Phys Material
None
Phys Material Mask
Material
Material Domain
Surface
Blend Mode
Opaque
Shading Model
Unlit
Two Sided
Use Material Attributes
Cast Ray Traced Shadows
Subsurface Profile
Advanced
Physical Material Mask


File
Edit
Asset
Window
Tools
Help
M_Ring
M_Ring1
Save
Browse
Apply
Search
Home
Hierarchy
Live Update
Clean Graph
Preview State
Hide Unrelated
Stats
Platform Stats
Perspective
Lit
Show
M_Ring1 > Material Graph
Details
Parameters
Search
Physical Material
Phys Material
None
Phys Material Mask
Material
Material Domain
Surface
Blend Mode
Translucent
Shading Model
Unlit
Two Sided
Use Material Attributes
Cast Ray Traced Shadows
Subsurface Profile
Advanced
Physical Material Mask
Particle Color
RadialGradientExponential
UVs (V2)
CenterPosition (V2)
Radius (S)
Density (S)
Invert Density (B)
Dynamic Parameter
Density
Radius1
Radius2
Param4
Subtract
Multiply
M_Ring1
Emissive Color
Opacity
World Position Offset
Refraction

Explanation of the process being used to generate the effect

The emitters, Ring and Ring001, can generate the outer fluctuating ring with glitter chips. Ring001 has a higher vertical fluctuation of the ring than Ring. The emitter, Fountain, can generate the inner horizontal halo.



Annotations/Descriptions for each of the emitters and modules used

Ring

The main body of the vertical torus. The explicit material is M_Ring.

1. Emitter Update
   Use Emitter State and Spawn Burst Instantaneous to control the loop time and the spawn count.
2. Particle Spawn
   Use Initialise Particle, ShapeLocation, Initial Mesh Orientation to control the location and size of the torus.
3. Particle Update
   Use Particle State, Solve Forces and Velocity and Dynamic Material Parameters to control the vertical fluctuation of the particles.
4. Render
   Use Mesh Renderer with PlaneMesh and M_Ring to generate the base shape.

Ring001

Ring001 is the copy of Ring. The decrease of spawn count in Emitter Update and the increase of wpo in Particle Update can result in more obvious vertical fluctuation than Ring.

Fountain

The main body of the horizontal torus. The explicit material is M_Ring1.

1. Emitter Update
   Use Emitter State, Spawn rate and Spawn Burst Instantaneous to control the loop time, the frequency and the spawn count.
2. Particle Spawn
   Use Initialise Particle and Align Sprite to Mesh Orientation to make the horizontal torus conform to the size and location of the Ring.
3. Particle Update
   Use Particle State, Scale Sprite Size, Solve Forces and Velocity, Dynamic Material Parameters and Scale Colour to control the animation of spread of the horizontal torus.
4. Render
   Use Sprite Renderer with M_Ring1 to generate the base shape.

● C++ Parameters Breakdown

The user exposed parameters are LifeTime and wpo. Both of them are floats.
The purpose of LifeTime is to control the loop duration of NS_GhostField. The maximum value is 5 and the minimum value is 2.
The purpose of wpo is to control the maximum height of the vertical fluctuation. It can help players realise whether the lifetime of the ghost field is long or short in visual effects.
When the player goes back to the ghost location and the ghost has already existed over half maximum lifetime, the lifetime and Wpo of the ghost field will be the maximum, otherwise their values will be the minimum.

```
UPROPERTY(Replicated,EditAnywhere)float LongGhostFieldLifeTime = 5;
UPROPERTY(Replicated,EditAnywhere)float ShortGhostFieldLifeTime = 2;
UPROPERTY(Replicated,EditAnywhere)float ShortWpo = 150;  Unchanged in assets
UPROPERTY(Replicated,EditAnywhere)float LongWpo = 300;  Unchanged in assets
```

```
if(CurrentGhostTime<=MaximumGhostTime/2)
{
    WhetherGhostFieldLifeTime = false;
}
else
{
    WhetherGhostFieldLifeTime = true;
}
```

```
void AA01_BlockoutSadiumCharacter::MulticastGhostField_Implementation()
{
    if(NS_GhostField)
    {
        UNiagaraComponent* GhostFieldComponent = UNiagaraFunctionLibrary::SpawnSystemAtLocation(GetWorld(), NS_GhostField,
            Location: FVector(this->GetActorLocation().X,this->GetActorLocation().Y, InZ: 0), this->GetActorRotation());
        if(WhetherFieldLifeTime)
        {
            GhostFieldComponent->SetNiagaraVariableFloat(FString( Str: "LifeTime"),LongGhostFieldLifeTime);
            GhostFieldComponent->SetNiagaraVariableFloat(FString( Str: "wpo"),LongWpo);
        }
        else
        {
            GhostFieldComponent->SetNiagaraVariableFloat(FString( Str: "LifeTime"),ShortGhostFieldLifeTime);
            GhostFieldComponent->SetNiagaraVariableFloat(FString( Str: "wpo"),ShortWpo);
        }
    }
}
```

NS_GravityField

- Overview of Effect

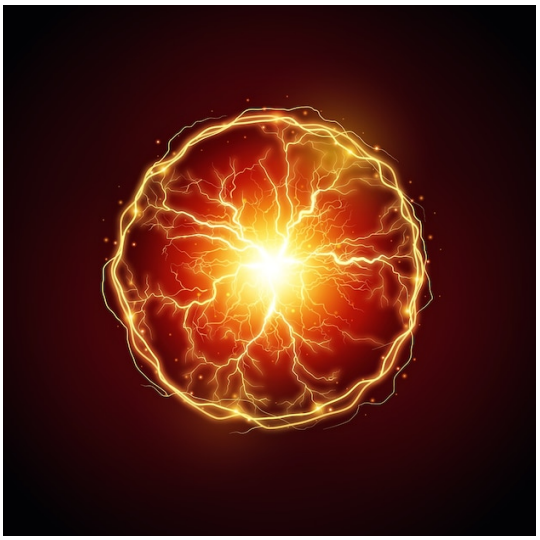
NS_GravityField is the particle effect to display the centre of the gravity field. Its purpose is to make the rival players notice the effective area of GravityField.

- Effect Description

Description of what the effect looks like

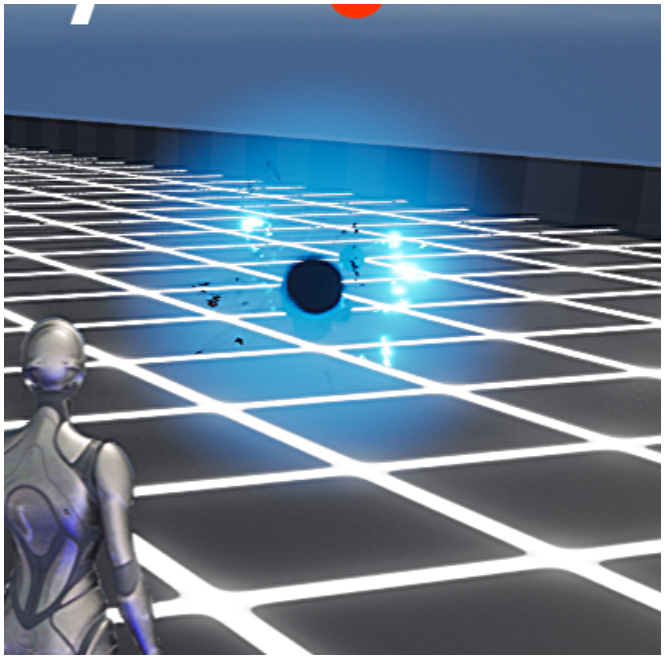
NS_Gravity Field looks like a lightning ball. The main body is black and it will keep releasing lightning inside the blue volume.

Reference images used in the creation of the effect



Reference Link:
https://www.freepik.com/free-photos-vectors/lightning-ball

Screenshots of effect in-game

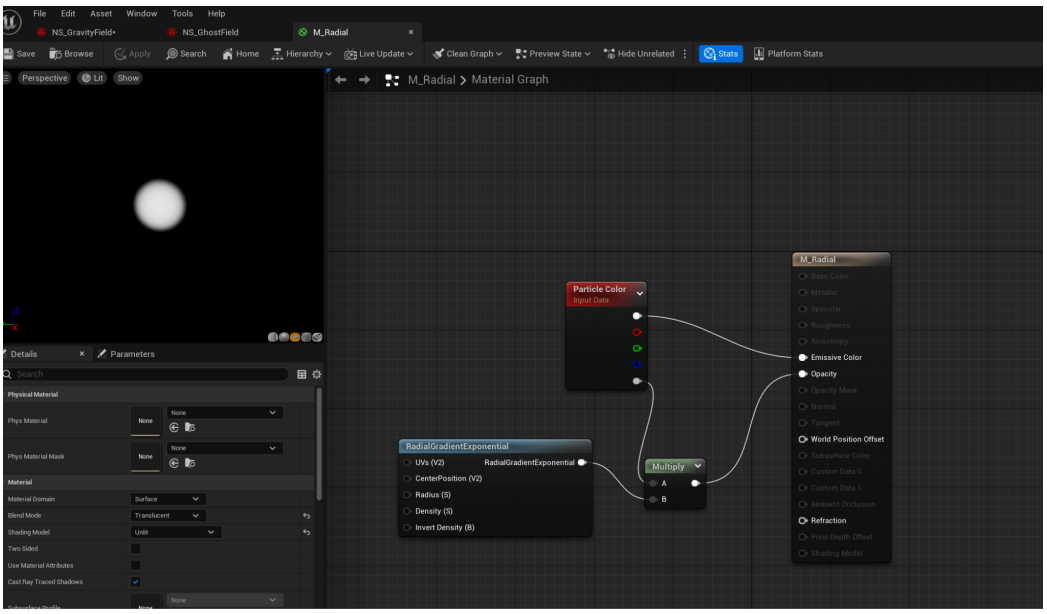


The properties used & Concrete values

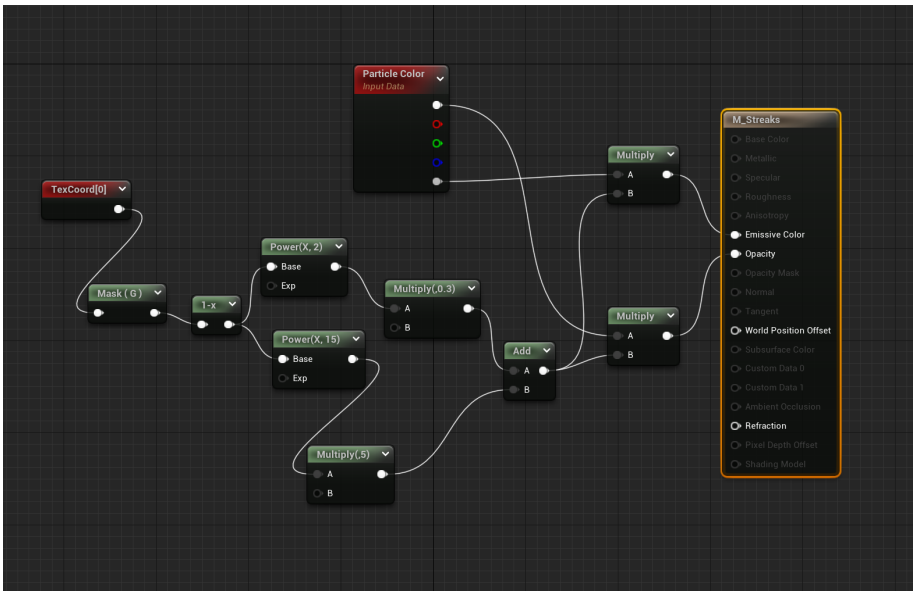
Max Pool Size is set to 32. When emitters finish, the Niagara system will be killed. The other properties are left as default. There are seven emitters. The user exposed parameters are LifeTime. It can impact the lifetime of the Niagara system. The materials I used are M_Radial, M_Lighting_Inst and M_Streaks.

- Niagara System / Emitters Breakdown

Screenshots of Niagara System and Emitter


File Edit Asset Window Tools Help
NS_GravityField•
NS_GhostField
M_Radial
Save Browse Apply Search Home Hierarchy Live Update Clean Graph Preview State Hide Unrelated Stats Platform Stats
Perspective Lit Show
M_Radial › Material Graph
Details
Parameters
Physical Material
Phys Material
Phys Material Mask
Material
Material Domain
Blend Mode
Shading Model
Two Sided
Use Material Attributes
Cast Ray Traced Shadows
Particle Color
Input Data
RadialGradientExponential
UVs (V2)
CenterPosition (V2)
Radius (S)
Density (S)
Invert Density (B)
Multiply
A
B
M_Radial
Base Color
Metallic
Specular
Roughness
Anisotropy
Emissive Color
Opacity
Opacity Mask
Normal
Tangent
World Position Offset
Subsurface Color
Custom Data 0
Custom Data 1
Ambient Occlusion
Refraction
Pixel Depth Offset
Shading Model


Search
Parameter Groups
Global Scalar Parameter Values
Max
Min
Save Sibling
Save Child
General
Phys Material
Parent
Lightmass Settings
Material Property Overrides
Advanced
Asset User Data
Previewing
Preview Mesh


Particle Color
Input Data
TexCoord[0]
Mask (G)
1-x
Power(X, 2)
Base
Exp
Power(X, 15)
Multiply(.0.3)
Multiply(.5)
Add
Multiply
Multiply
M_Streaks
Base Color
Metallic
Specular
Roughness
Anisotropy
Emissive Color
Opacity
Opacity Mask
Normal
Tangent
World Position Offset
Subsurface Color
Custom Data 0
Custom Data 1
Ambient Occlusion
Refraction
Pixel Depth Offset
Shading Model

Explanation of the process being used to generate the effect

The emitters, Core and Core001, generate the core of the dark ball and its volatility. Glow can generate the blue glow around the core. Lightning and Lightning_Dark can generate the bright blue and dark lightning toruses which spawn from the core randomly. Streaks and Streaks_dark can generate tiny sparks around the blue glow.

Annotations/Descriptions for each of the emitters and modules used

Core

The core of the dark ball

1. Emitter Update
   Use Emitter State and Spawn Rate to control the loop time and the frequency to spawn the core.
2. Particle Spawn
   Use Initialise Particle, ShapeLocation, AddVelocity to control the location and the floating of the core.
3. Particle Update
   Use Particle State and Scale Colour to control the colour of the core.
4. Render
   Use Sprite Renderer with M_Radial to generate the base shape.

Core001

Core001 is the copy of Ring. The change in the maximum velocity speed of AddVelocity can increase the unstable floating of the core.

Glow

The blue glow around the core

Glow is also a copy of Ring.

1. Particle Spawn
   The change in the colour and the increase in the Sprite Size Mode of InitializeParticle can make the blue glow wrap around the core.

Lightning & Lightning_Dark

The bright blue and dark lightning toruses which spawn from the core randomly

1. Particle Update
   Use Particle State, Update Mesh Orientation, Scale Colour, Solve Forces and Velocity, Dynamic Material Parameters and Align Sprite to MeshOrientation to control the animation of spread of the lightning around the core.
2. Render
   Use Sprite Renderer with M_Lighting_Inst to generate the base shape to relatively semi torus.

Streaks & Streaks_dark

Tiny bright blue and dark sparks around the blue glow

1. Particle Spawn
   Use Initialise Particle, ShapeLocation, AddVelocity to control the location and the motion speed of the sparks.
2. Render
   Use Sprite Renderer with M_Streaks to generate the base shape to relatively strip.

● C++ Parameters Breakdown

The user exposed parameters are Lifetime. It is a float.

The purpose of Lifetime is to control the loop duration of NS_GravityField. The maximum value is 10 and the minimum value is 5.

When the player presses E again to spawn the gravity field and the gravity field bullet has already existed over 2.5, the lifetime of the gravity field will be the maximum, otherwise its lifetime will be the minimum.


```
{
    if(NS_Field)
    {
        UNiagaraComponent* FieldComponent = UNiagaraFunctionLibrary::SpawnSystemAtLocation(GetWorld(), NS_Field,
            FieldLocation, this->GetActorRotation());
        if(WhetherFieldLifeTime)
        {
            FieldComponent->SetNiagaraVariableFloat(FString(Str: "Lifetime"),LongFieldLifeTime);
        }
        else
        {
            FieldComponent->SetNiagaraVariableFloat(FString(Str: "Lifetime"),ShortFieldLifeTime);
        }
    }
}
```

```
    SpawnedField->Parent = this;
    if(Cast<AGravityFieldBullet>(Actor)->CurrentLifetime<=2.5)
    {
        WhetherFieldLifeTime = false;
        SpawnedField->CurrentLifetime=ShortFieldLifeTime;
    }
    else
    {
        WhetherFieldLifeTime =true;
    }
    MulticastLaunchField();
```

```
bool WhetherSpawnBullet = false;
UPROPERTY(Replicated,EditAnywhere)bool WhetherFieldLifeTime;
UPROPERTY(Replicated,EditAnywhere)float LongFieldLifeTime = 10;
UPROPERTY(Replicated,EditAnywhere)float ShortFieldLifeTime = 5;
```

Chaos Destruction

Geometry Collection — GC_FragilePillar

● Overview of Effect

GC_FragilePillar is a pillar that can be broken into many pieces. There are a total 4 pillars placed at each corner of the game map. Its purpose is to make players have a high and clear target to launch the hook when it is intact. Also, players can destruct it to interfere with the motion of the ball.

● Effect Description

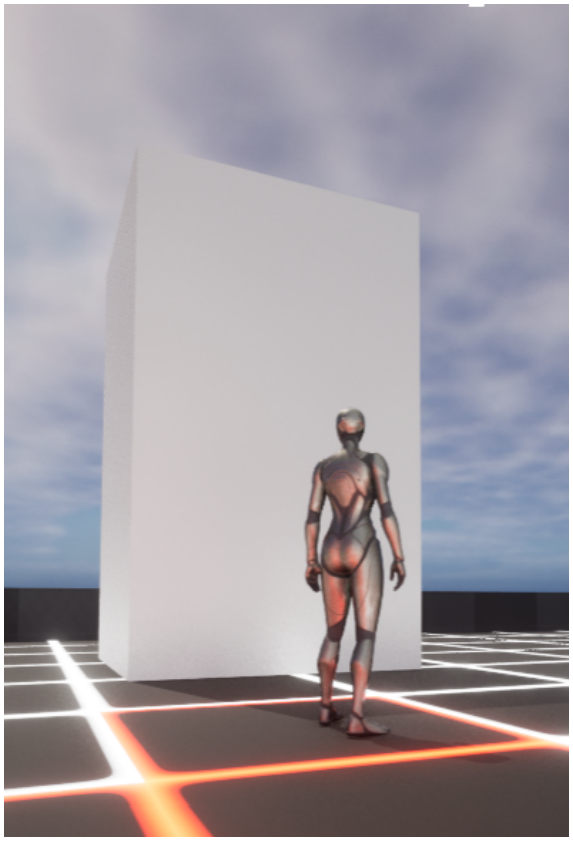
Description of what the Geometry Collection looks like

GC_FragilePillar looks like a big white cuboid and it can be divided by several irregular fragments.

Reference images used in the creation of the effect

Reference Link:
https://www.exhibitionsupply.com/product/22288/11-12-x-11-12-12/?origin=google_product_ads&gclid=CjwKCAjwzNOaBhAcEiwAD7Tb6A8S_sBhuWHNqRagFCD_RYlTyLEW-jaDZOEYHxmXStExnxx4angUpxoCMmcQAvD_BwE

Screenshots of effect in-game



The properties used & Concrete values

The Fracture types used are two uniforms. The numbers of nodes in level 1 and level 2 are 150 and 1200 respectively. The min voronoi sites and max voronoi sites are set to 20. The other properties are left as default.

Anchor Field

● Overview of Effect

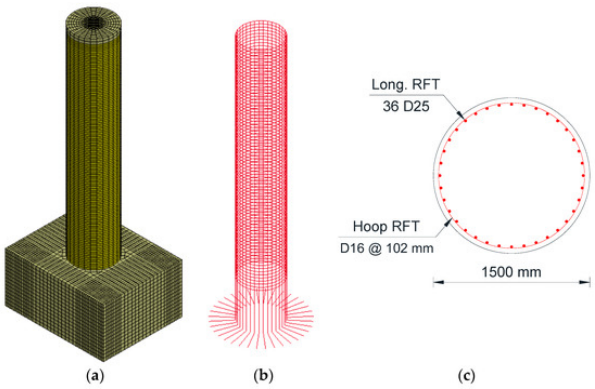
This field is applied in GC_FragilePillar. Its purpose is to prevent the pillar instantly starting falling apart at the beginning of the game. Players will not be disturbed by pillar fragments when they do not want to destroy the pillar.

● Effect Description

Description of what the layout of the Anchor Fields look like

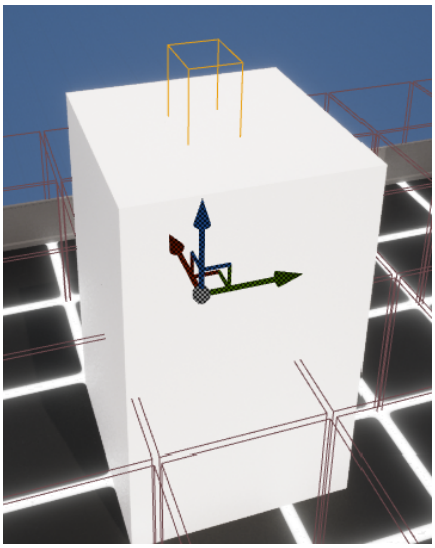
The layout of anchor fields looks like the column core of the pillar. It can be regarded as the gravity centre of the pillar to maintain the pillar foundation form.

Reference images used in the creation of the destruction controlling effect



Reference Link:
https://www.mdpi.com/2073-4360/8/12/432

Screenshots of effect in-game



The properties used & Concrete values

BP_AnchorField is a transient field. Culling Field is set to BoxFalloff, UniformInteger and Field_Culling_Outside. Falloff is set to BoxFalloff with magnitude(1), min range(0), max range(1). UniformInteger is set to 3.

Other Physics Field - BP_Bomb

- Overview of Effect

The player can throw out a bomb. Its purpose is to help players have an efficient method to destroy the pillar when they want to interfere with the ball.

- Effect Description

Description of what the desired Physics Field interaction looks like

The player will throw out a sphere in a parabola. And the sphere will cause an explosion to impact and destroy GC_FragilePillar when it hits the pillar.

Reference images used in the creation of the effect

Reference Link:
https://en.wikipedia.org/wiki/Explosion

Screenshots of effect in-game

The properties used & Concrete values

BP_Bomb is a transient field. Culling Field is set to Field_Culling_Outside. Falloff is set to Radial Falloff. The Strain Amount is set to 500000. Falloff Shape is a sphere and its radius is 0.25, 0.25, 0.25 scale of a default sphere.

Physics Constraints

A Wrecking Ball

- Overview of Interaction

It will cause the wrecking ball to move max 2 metres in any linear direction. Its purpose is to increase the difficulty of controlling the football when the rival players use it to interfere. The physics constraints can restrict the wrecking ball's sphere of influence and prevent it interfering with players and the motion of football anywhere on the game map.

- Overview of Interaction

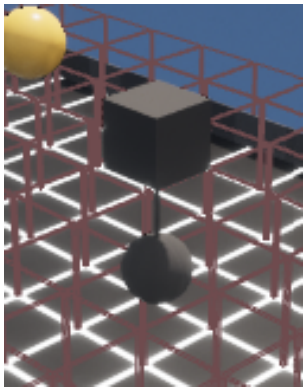
Description on how the physics interaction

The ball can move 2 metres in any linear direction, which can form a sphere with a radius of 2 metres.

Reference images used in the creation of the effect

Reference Link:
https://www.cdrecycler.com/article/wrecking-ball-malfunctions-in-california/

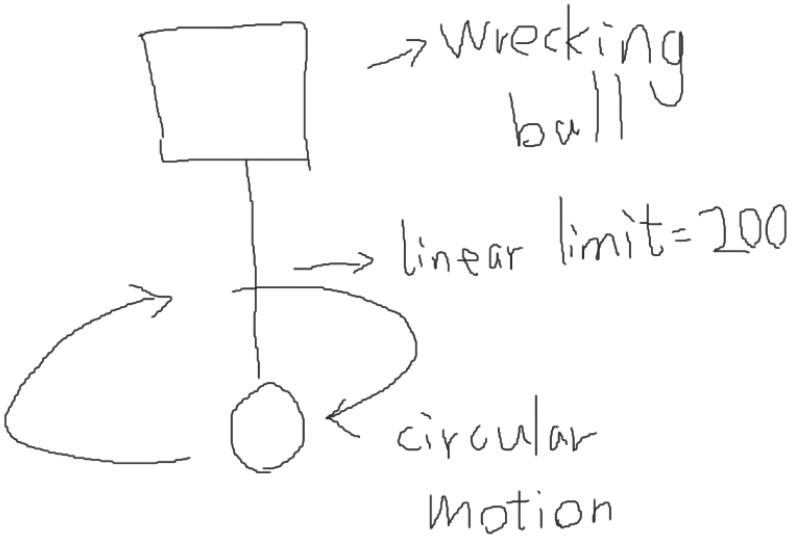
Screenshots of effect in-game



The properties used & Concrete values

The mesh is a combination of one top square, one sphere and one cable. All X Motion, Y Motion and Z Motion are limited. The linear limit is set to 200. The Motor strength is left as default.

- Diagram of Interaction



Advanced Particle Effects

Destruction Aware Particle Effect

- Overview of Effect

It is a black smoke. Its purpose is to cooperate with the trailing fire when the pillar fragments fall. This can increase player engagement.

- Effect Description

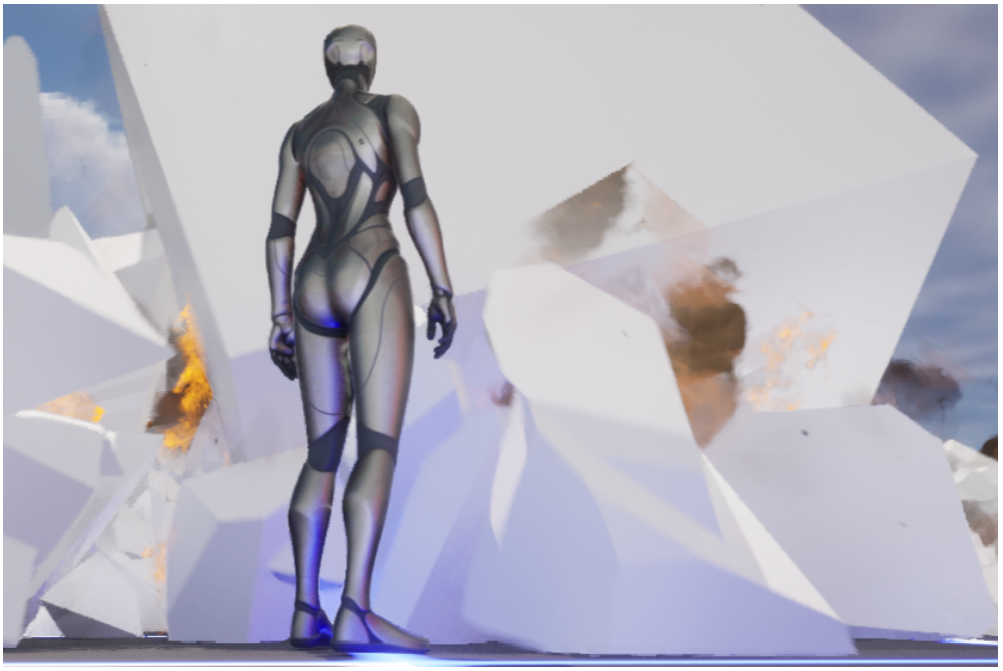
Description of what the effect looks like

The effect looks like a small cloud of black smoke caused by the fire.

Reference images used in the creation of the effect

Reference Link:
https://kmph.com/news/local/plastic-fire-sends-large-plume-of-smoke-into-the-air

Screenshots of effect in-game



The type of destruction data being listened for

The Data Source is set to Trailing Data. Data Process Frequency is set to 20 and Min Speed to Spawn Trailing Particles is set to 10 to prevent excessive black smoke generation.

The Geometry Collections outputting destruction data with an explanation why these Geometry Collections were chosen

This effect is applied to GC_FragilePillar. When the pillar is destroyed and its fragments fall, there will be several clouds of black smoke generated with the trailing fire. This can help the destruction of the fragile pillar become more real.

Collision Enabled Particle Effect

- Overview of Effect

It is a tiny ellipsoid bullet that can be shot by the player. Its purpose is to help players have a long-distance method to destroy the pillar a bit.

- Effect Description

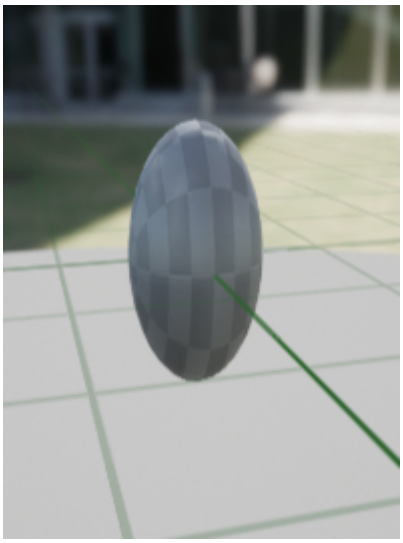
Description of what the effect looks like

It is a tiny ellipsoid bullet that can be shot by the player and can rebound.

Reference images used in the creation of the effect

Reference Link:
https://123sonography.com/ebook/left-ventricle

Screenshots of effect in-game



The values used in the collision module of the emitter

Radius Calculation Type is set to Sprite. The particle radius scale is 1.0. The restitution is set to 0.6. The Friction is set to 0.25 and Friction During a Bounce is set to 0.0.

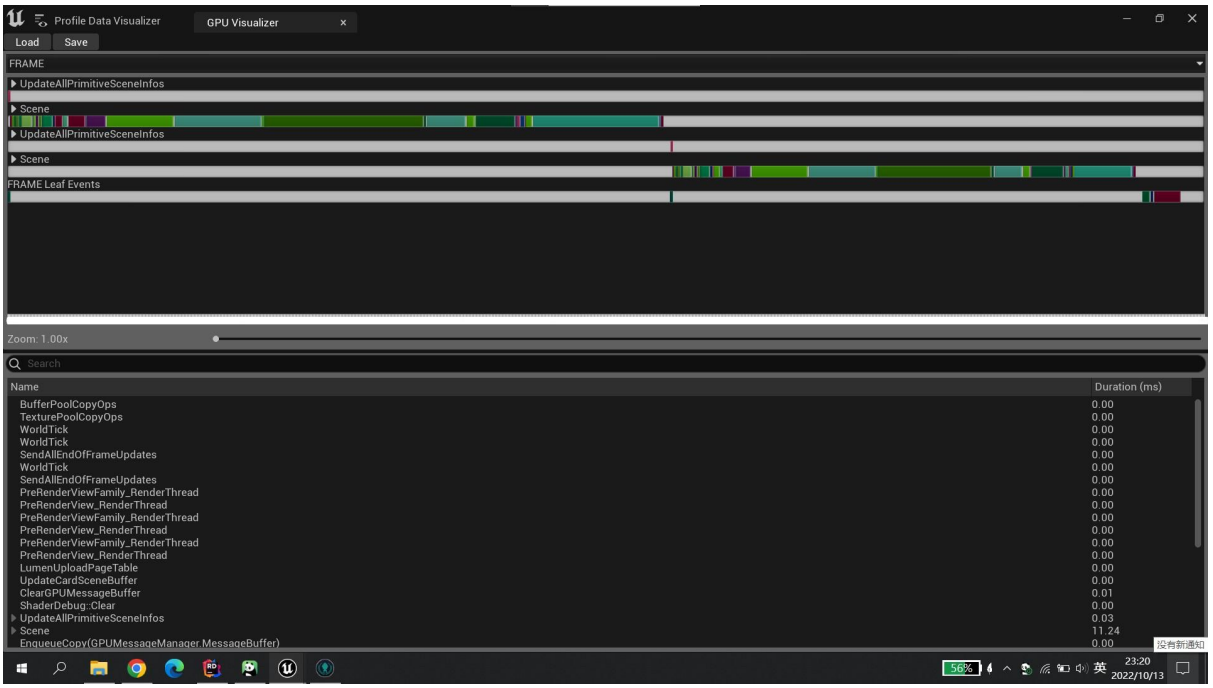
Optimisation

Statistics Auditor Report

Object	Actor(s)	Type	Count	HWInstanc	Inst Sectior	Tris	Sum Tris	Size	VC	Inst VC	Avg LM	Avg OL	Sum Avg	Cost	LM	Res	Min R	Max R	Avg R
			548	548	548	5,164	8,868	430.958	0 KB	0 KB	0	0	0	0	641.03	4,996.5	27.386	1,638,400	245,297.8
SM_SkySphere	StaticMeshActor_UAID_	StaticMesh	1	1	1	3,968	3,968	202.751	0 KB	0 KB	0	0	0	0	2.659 KI	32	1,638,400	1,638,400	1,638,400
Sphere	2 Actors	StaticMesh	2	2	2	960	1,920	113.404	0 KB	0 KB	0	0	0	0	0 KB	64	50	50	100
SM_Corner	4 Actors	StaticMesh	4	4	4	132	528	43.374 K	0 KB	0 KB	0	0	0	0	42.559 K	256	196.595	220.843	859.124
Cube	4 Actors	StaticMesh	5	5	5	48	240	20.782 K	0 KB	0 KB	0	0	0	0	0 KB	320	27.386	380.994	926.474
SM_Straight	54 Actors	StaticMesh	54	54	54	22	1,188	19.265 K	0 KB	0 KB	0	0	0	0	574.541	3,456	148.924	223.194	8,634.161
SM_Floor	2 Actors	StaticMesh	2	2	2	32	64	18.641 K	0 KB	0 KB	0	0	0	0	21.279 K	128	141.421	141.421	282.843
Plane	480 Actors	StaticMesh	480	480	480	2	960	12.741 K	0 KB	0 KB	0	0	0	0	0 KB	30,720	141.421	141.421	67,882.41

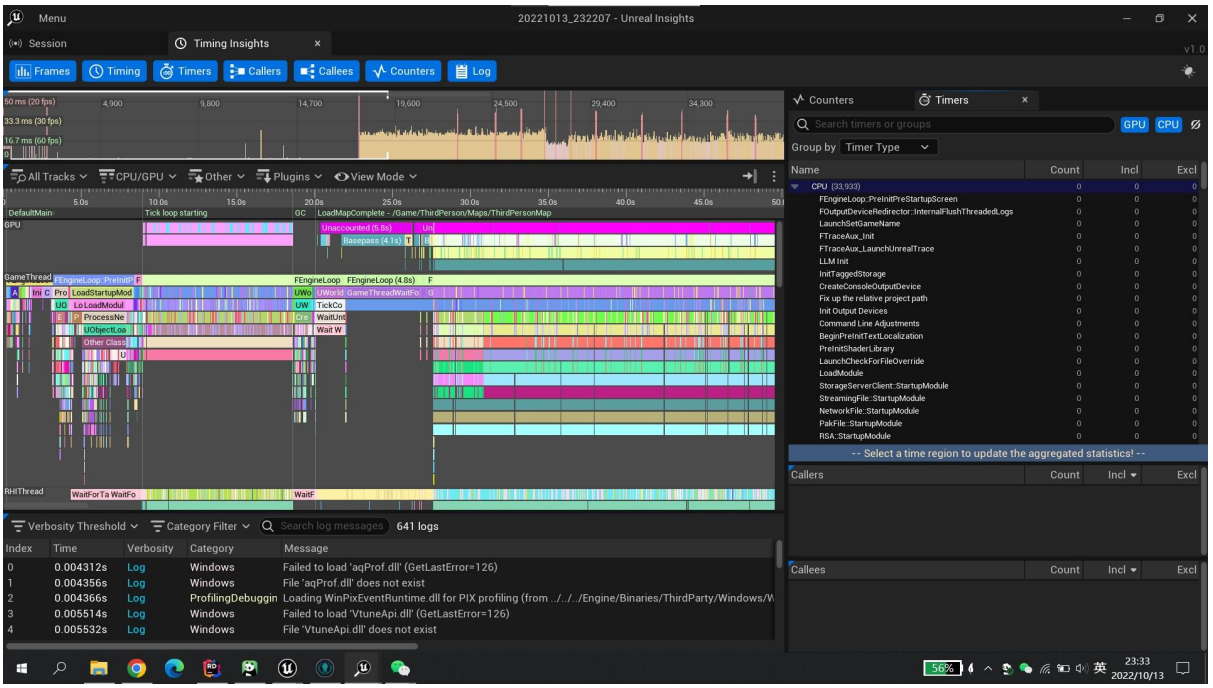
There are 480 planes, which are the floors in the game map. And each plane contains 2 triangles. The sum of triangles contained in all planes is 960, which is much lower than the triangles contained in SM_SkySphere. Also, the resource size of those planes is only about 12 KB. Therefore, the planes are in an acceptable range and do not need more optimization. According to the previous update of map, all SM_Floor should be replaced by Plane. There are 2 missing SM_Floor that have 18KB resource size. I will delete them to reduce the load. In addition, I will decrease the number of SM_Straight by increasing the length of a single SM_Straight. This solution can decrease the sum of triangles contained in all SM_Straight.

GPU Profiler Report



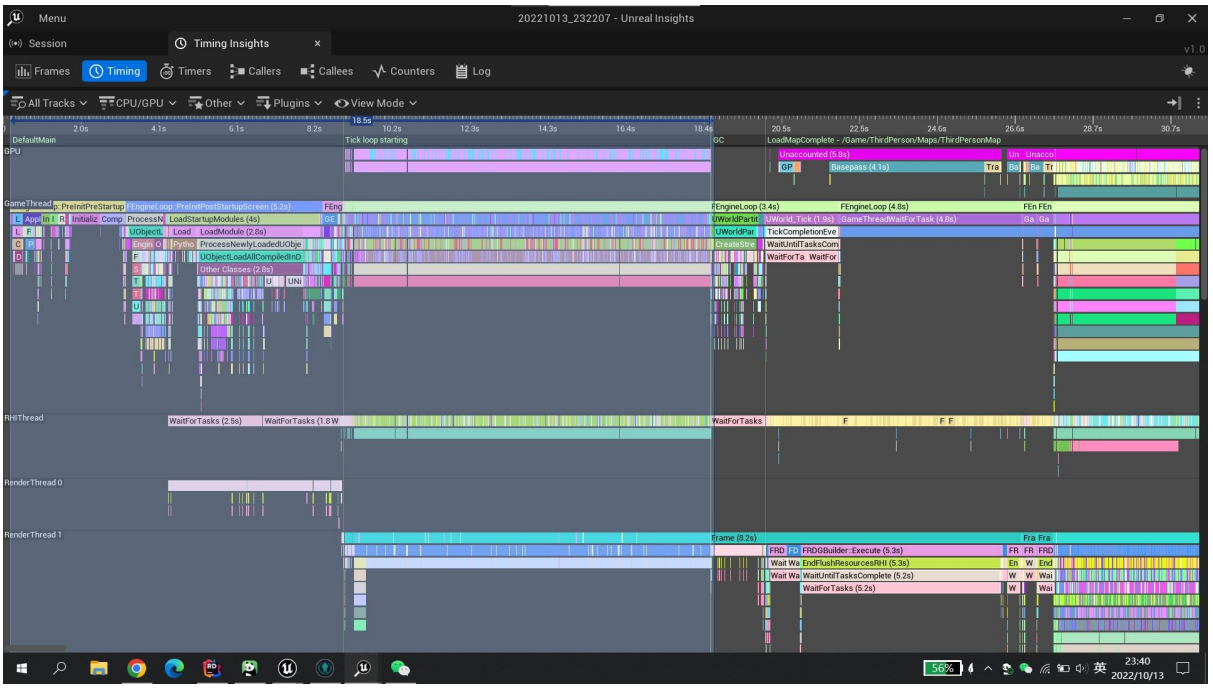
The duration to load the scene is 11.24ms. And the subsequent duration to update the scene is about 16ms. The other durations are mainly lower than 0.05ms. Therefore, I should focus more on the scene and its update during the game. Reducing the triangles can decrease the duration of loading the scene, which improves the performance.

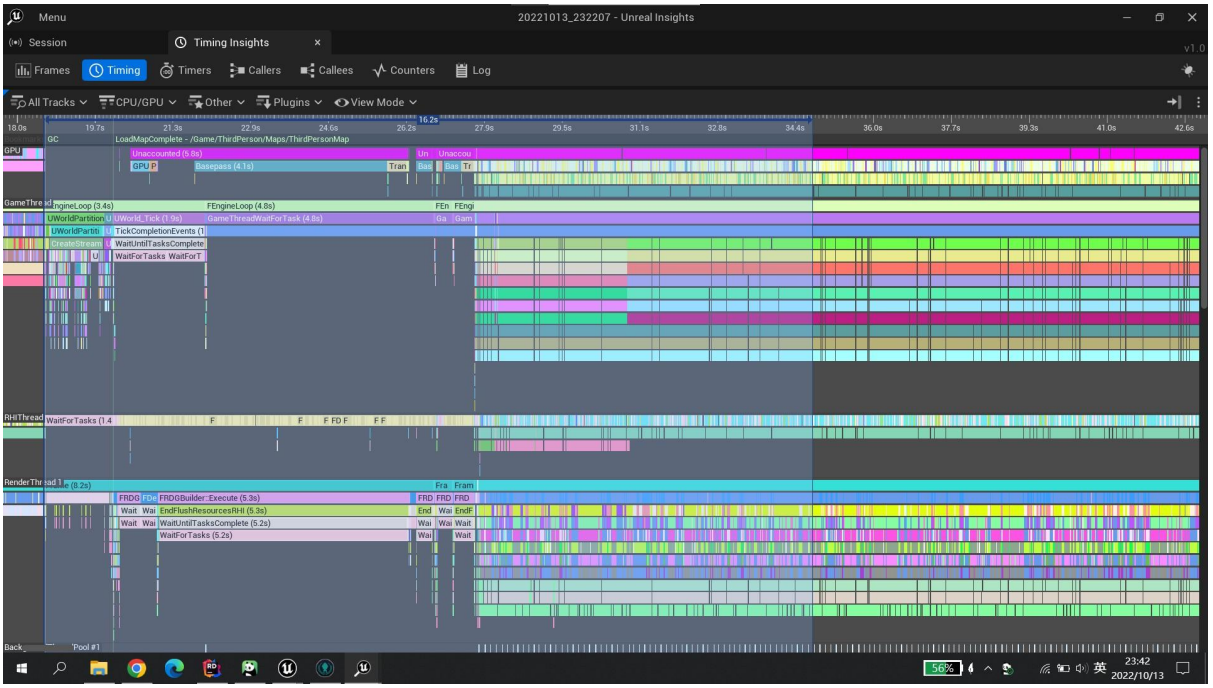
Unreal Insights Report



According to the frame panel, the game screen is always around 30 frames per second. Although 30 FPS is not high, this game should focus more on the stability of frame considering the game is a multiplayer sport game. I need to reallocate the resources that need to be loaded or be updated per second, such as Tick(). This can help the game screen have a more stable FPS, which also improves the performance.

Timing Sections Report 1 & 2





The Timing panel shows individual trace events and provides time measurements with sub-microsecond precision. The current screenshots do not present a seriously weird time period. I will record the time duration to load the particle effects in the future test to analyse the resources needed in those particle effects, for example, the gravity field and its bullet. This can provide more details to improve the performance.

Lighting

● Overview of Effects

Every 6 Point Lights can be regarded as LED light. Rect Light can be regarded as the main light sources of the whole game map. Spot Light can be regarded as a projection lamp. Their purpose is to increase players' immersion in the game and the colours of spotlights help players recognize which goal belongs to which team.

● Effect Description

Description of what the effect looks like

The combination of 6 Point Lights looks like a strip of LED lights with different colours. Each rect light looks like the ambiance light in home with soft yellow light. The spot light looks like the projection lamp with the corresponding colour of each team.

Reference images used in the creation of the effect

Reference Link:
https://www.ledsupply.com/blog/what-you-need-to-know-about-leds/

Reference Link:
https://www.amazon.com/Balkwan-Projection-Rotation-Rainbow-Romantic/dp/B08XTXM2HS/ref=asc_df_B08XTXM2HS/?tag=hyprod-20&linkCode=df0&hvadid=507830692786&hvpos=&hvnetw=g&hvrand=3451945561604482942&hvpone=&hvptwo=&hvqmt=&hvdev=c&hvdvcmdl=&hvlocint=&hvlocphy=1027744&hvtargid=pla-1214974951368&psc=1

Reference Link:
https://www.istockphoto.com/photo/blue-spotlight-on-stage-performance-gm1187168424-335242802

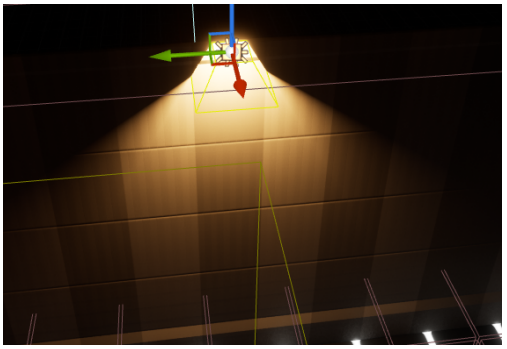
Screenshots of effect in-game

Point Light

Be placed at the top of the goal to notify players.

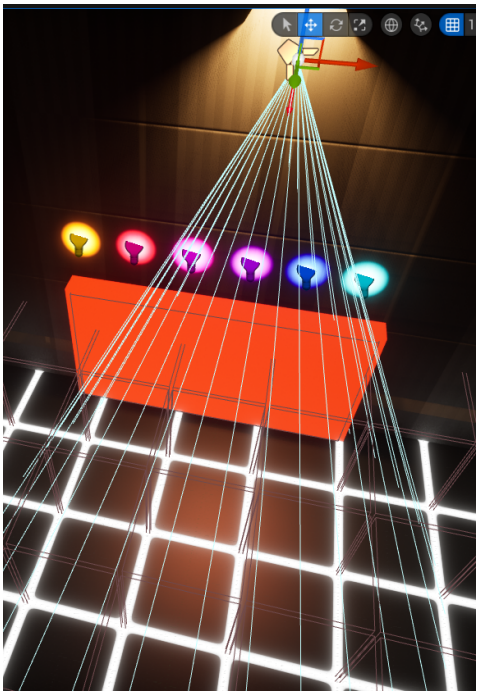
Rect Light

Be placed at the top of the walls to provide enough light sources to the map.



Spot Light

Be placed in front of the goals to help players recognize.



The properties used & Concrete values

Point Light

The attenuation radius is about 47. The temperature is 4500. The intensity is 36. Those values can make the point light clear and not dazzling. The light colours have yellow, red, pink, purple, blue and cyan.

Rect Light

The source width is 75. The source height is 64. The temperature is 3000, which makes it closer to real light temperature. The intensity is 100 that ensures it is bright enough. The light colour is white.

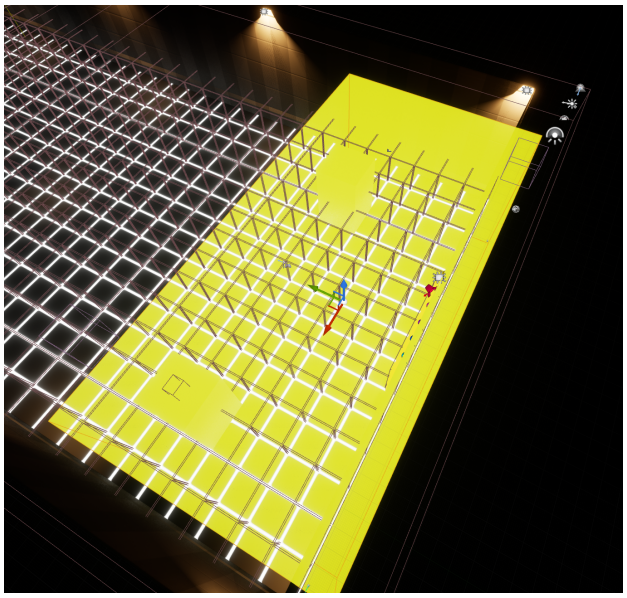
Spot Light

The outer cone angle is 27. The intensity is 160. The attenuation radius is about 2000 that ensures it can be clear at the floor. The light colours are red and blue.

• Screenshots / Settings Breakdown

The lightmass importance volume(s) used

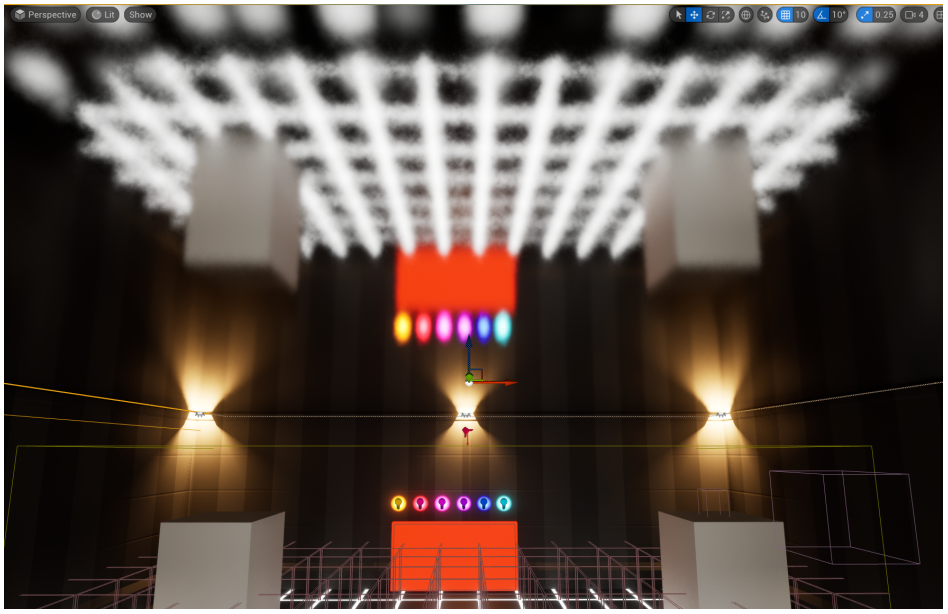
Two lightmass importance volumes are placed at the front of the goals to highlight the fighting before the goals.



Discussion of the settings used for the lightmass importance volume(s)

The scale is X:13.5, Y:6.75 and Z:5.5, which ensures the volume can cover enough front place. Shaded Volume Opacity Value is 0.5, which makes the light mass suitable enough.

Discussion of Reflection captures used



Two planar reflections are placed at the ceilings of each side and do not impact the middle part of the game map. The reflections can increase the immersion of indoor soccer. And the middle part does not need reflections due to less light and the consideration of game performance.

MetaSound

● Overview of Interaction

MS_Goal is a sound for congratulations when one team scores. The repeat speed will change when the score of one team exceeds the other team's score. Its purpose is to increase the immersion and provide an audio reward and feedback to the player.

● Effect Description

Description on how the MetaSound effect is used within game

MS_Goal can be regarded as applause. It will start when one team scores and last for 3 seconds. The repeat speed of the sound will increase when the score of one team exceeds the other team's.

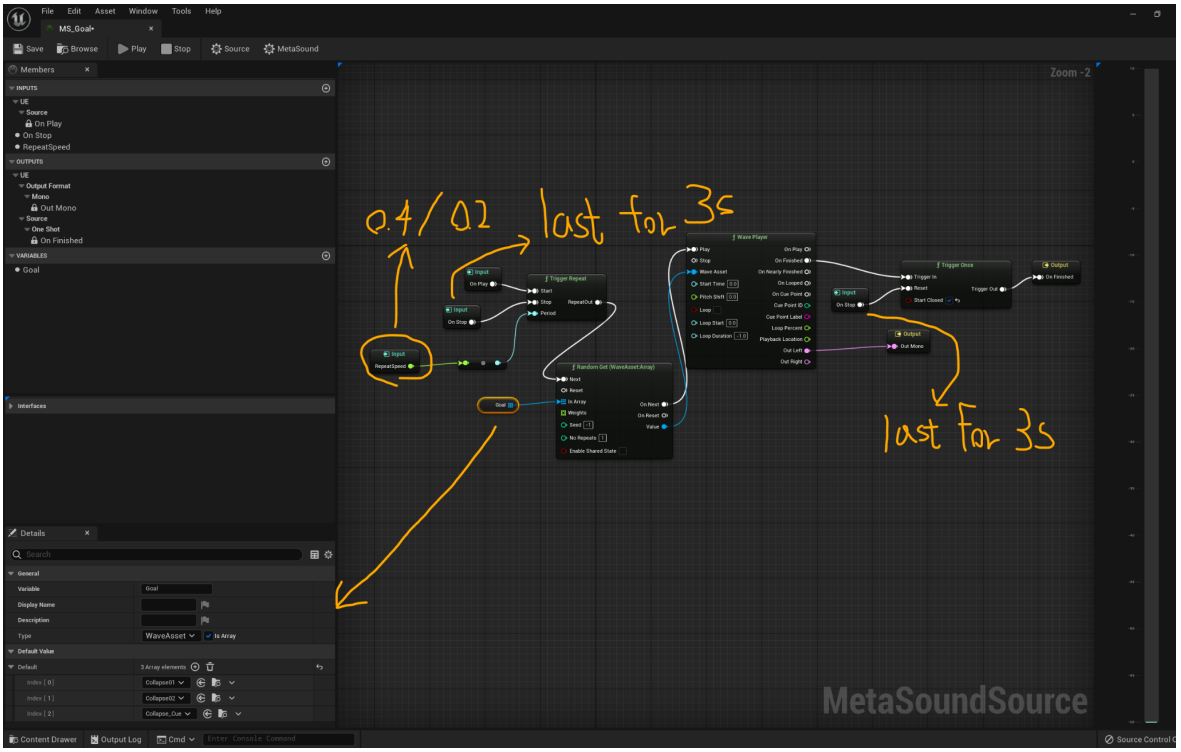
Reference images / sound effects used in the creation of the effect (if any)

Reference Link:
https://www.123rf.com/photo_83367509_human-hands-clapping-applause-clap-vector-illustration.html

The properties used & Concrete values

The inputs include On Play and On Stop. On Play depends on OverlapBegin of the goal. On Stop depends on Tick. The variable created is Goal and has WaveAsset type. The sound waves choosed are the array of 3 Collapse.

• Diagram Breakdown



The values of Repeat Speed are 0.4 and 0.2.

On Stop can control the duration of MS_Goal. The duration is 3 seconds.

The node Goal has WaveAsset Type. Its array is Collapse01, Collapse02 and Collapse_Cue. Through RandomGet, the array sounds like regular applause.