# Bajaj Institute of Technology, Wardha

## Role of C++ in Unreal Engine for Game Development

NAME - SHREEYOG SHENDE

## *Abstract/Introduction*

C++ is a powerful and versatile programming language that was created by Bjarne Stroustrup, a Danish computer scientist, back in 1979. Originally, it was an extension of the C language and was known as C with Classes. What makes C++ special is that it combines the best of both worlds; the low-level control you get from C with the high-level features you might find in more modern languages.

## *Objectives*

- **Maximize Performance:**
- Fine-tune game performance with low-level control.
- **Create Custom Mechanics:**
- Build unique game logic and systems.
- **Customize the Engine:**
- Modify and extend Unreal Engine features.
- **Optimize Critical Systems:**
- Ensure fast, efficient processing for physics, AI, and animations.
- **Build Custom Tools:**
- Develop tools and editor extensions to streamline workflows.
- **Integrate Extra Tech:**
- Add third-party libraries and middleware.
- **Develop Core Systems:**
- Implement essential game systems like player controls and game modes.
- **Support Multiple Platforms:**
- Develop games for PC, consoles, and mobile with consistent quality.
- **Efficient Memory Management:**
- Avoid memory leaks and optimize resource usage.
- **Modular Design:**
- Create flexible, reusable game components.

## *Algorithms & Methods*

**1.Set Up Project:**
Create a new Unreal Engine project with C++ support.
**2.Define Classes:**
Write C++ classes for custom game elements (e.g., characters, objects).
**3.Implement Logic:**
Develop gameplay mechanics and logic in C++ code.
**4.Integrate with Blueprints:**
Expose C++ functions and properties to Unreal Engine's Blueprint system.
**5.Optimize Performance:**
Use efficient algorithms and manage memory effectively.
**6.Test & Debug:**
Test the game and debug using Unreal Engine's tools.
**7.Build & Deploy:**
Compile the project and deploy it to target platforms.

## *Diagram*

```
// Called to bind functionality to input
void AShooterCharacter::SetupPlayerInputComponent(UInputComponent*
PlayerInputComponent)
{
    //input player movement
    Super::SetupPlayerInputComponent(PlayerInputComponent);

    PlayerInputComponent->BindAxis(TEXT("MoveForward"), this,
&AShooterCharacter::MoveForward);
    PlayerInputComponent->BindAxis(TEXT("LookUp"), this,
&APawn::AddControllerPitchInput);
    PlayerInputComponent->BindAxis(TEXT("MoveRight"), this,
&AShooterCharacter::MoveRight);
    PlayerInputComponent->BindAxis(TEXT("LookRight"), this,
&APawn::AddControllerYawInput);
    PlayerInputComponent->BindAction(TEXT("Jump"),
EInputEvent::IE_Pressed, this, &ACharacter::Jump);

}

//forward and backward movement
void AShooterCharacter::MoveForward(float AxisValue)
{
    AddMovementInput(GetActorForwardVector() * AxisValue);
}

//left and right movement
void AShooterCharacter::MoveRight(float AxisValue)
{
    AddMovementInput(GetActorRightVector() * AxisValue);
}
```

Figure 17. Movement controle code

Fig. Movement control code

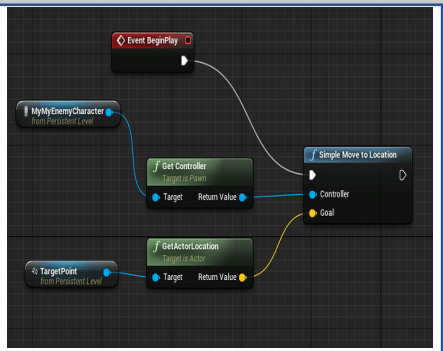


Fig. blueprint

## *Conclusion/Future Scope*

- **High-Performance Computing:**
- C++ will continue to be crucial for optimizing game performance, handling complex calculations, and ensuring efficient use of system resources.
- **AI and Machine Learning:**
- Future developments will leverage C++ for more sophisticated AI systems and machine learning models, enhancing gameplay with intelligent, adaptive behaviors.
- **Next-Gen Graphics and Rendering:**
- C++ will support advancements in graphics technology, including real-time ray tracing and high-fidelity visual effects, pushing the boundaries of realism in games.
- **Extended Platform Compatibility:**
- As new platforms and devices emerge, C++ will enable Unreal Engine to adapt, supporting a wider range of hardware, including next-gen consoles and advanced VR/AR systems.
- **Enhanced Networking and Multiplayer:**
- C++ will improve networking capabilities, making multiplayer experiences more seamless, scalable, and robust, with better synchronization and reduced latency.
- **Custom Tool Development:**
- Developers will use C++ to create custom tools and plugins, enhancing the Unreal Engine editor and streamlining development workflows.
- **Integration with Emerging Technologies:**
- C++ will facilitate the integration of new technologies, such as blockchain and cloud computing, into Unreal Engine, expanding the possibilities for game features and services.

## *References*

=>epic developer community
⇒ Google scholar
⇒ Unreal engine forums
⇒ Wikipedia
⇒ Unreal engine website