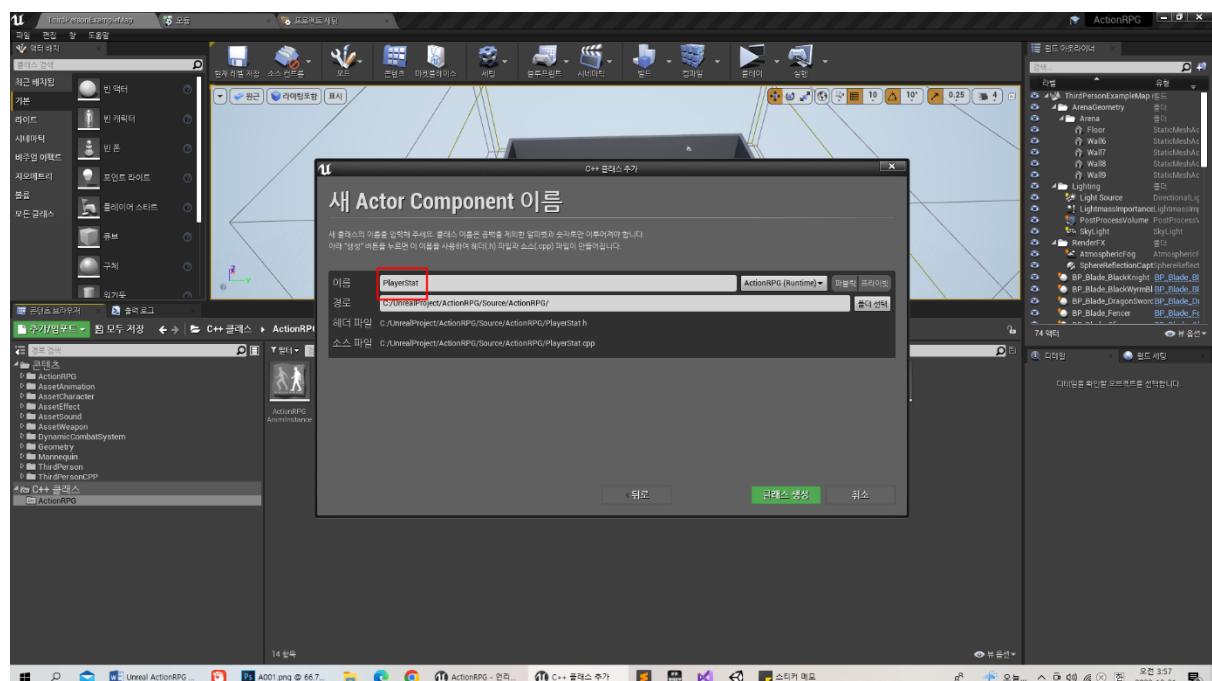
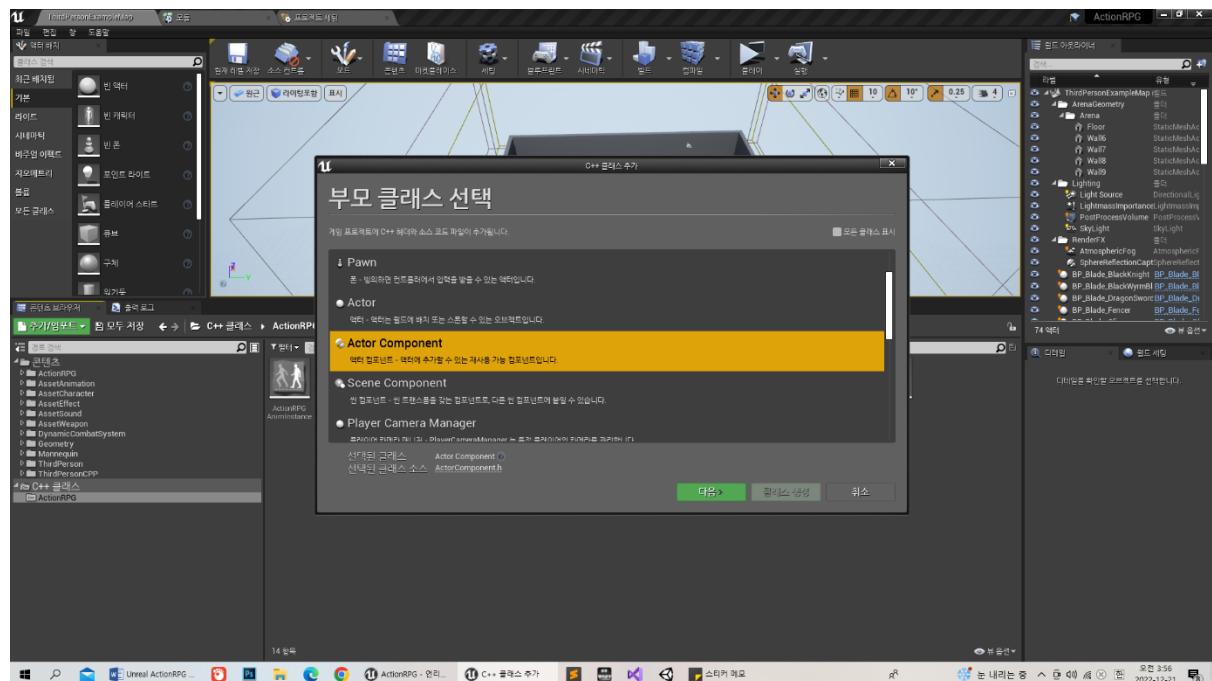
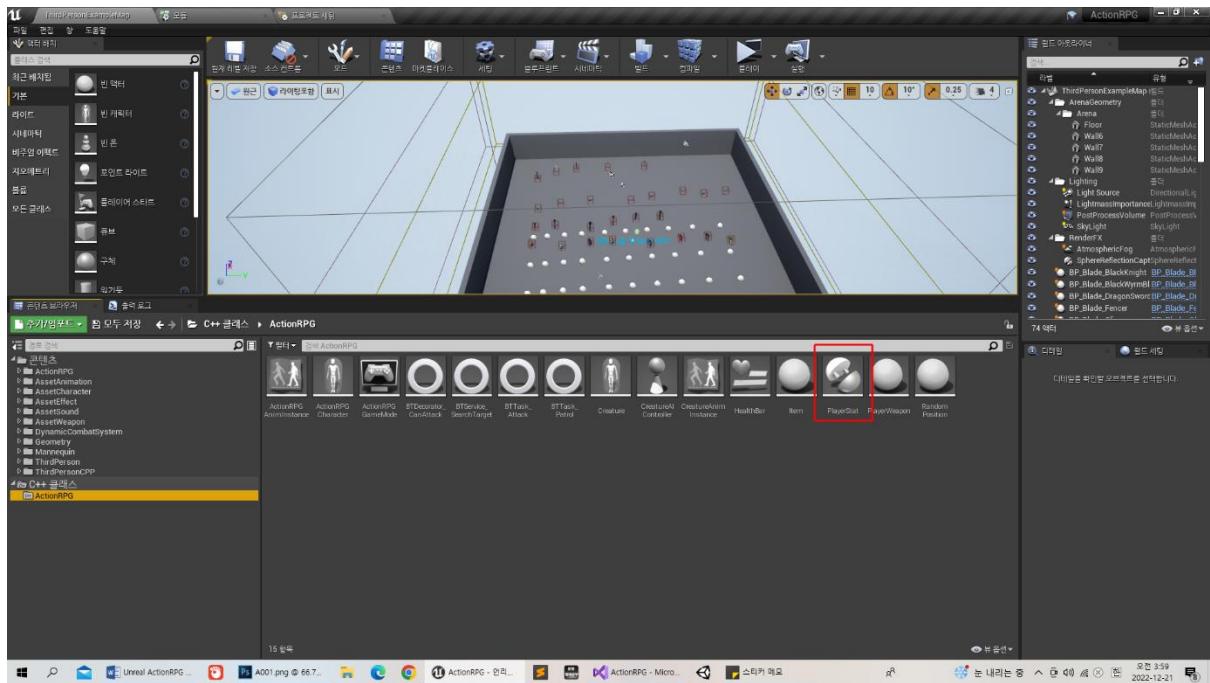


수치값을 코드에 넣는 것을 하드코딩이라고 합니다. 하드코딩을 없애주도록 합니다.

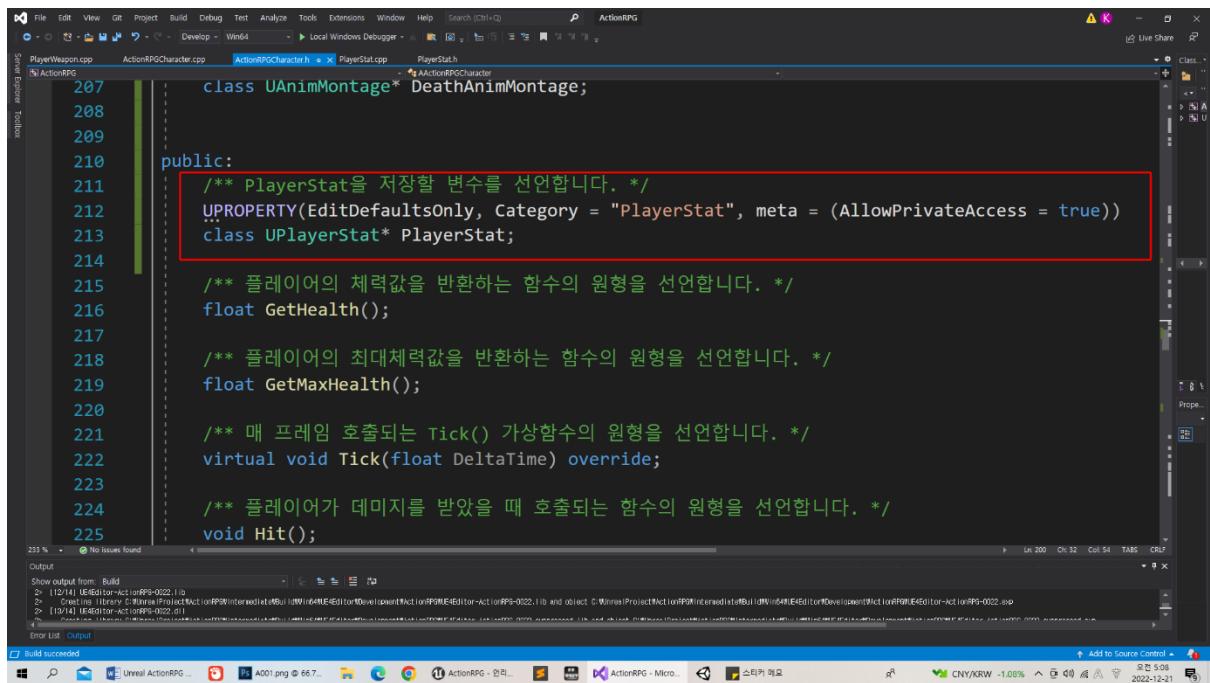
플레이어부터 해 주도록 합니다.

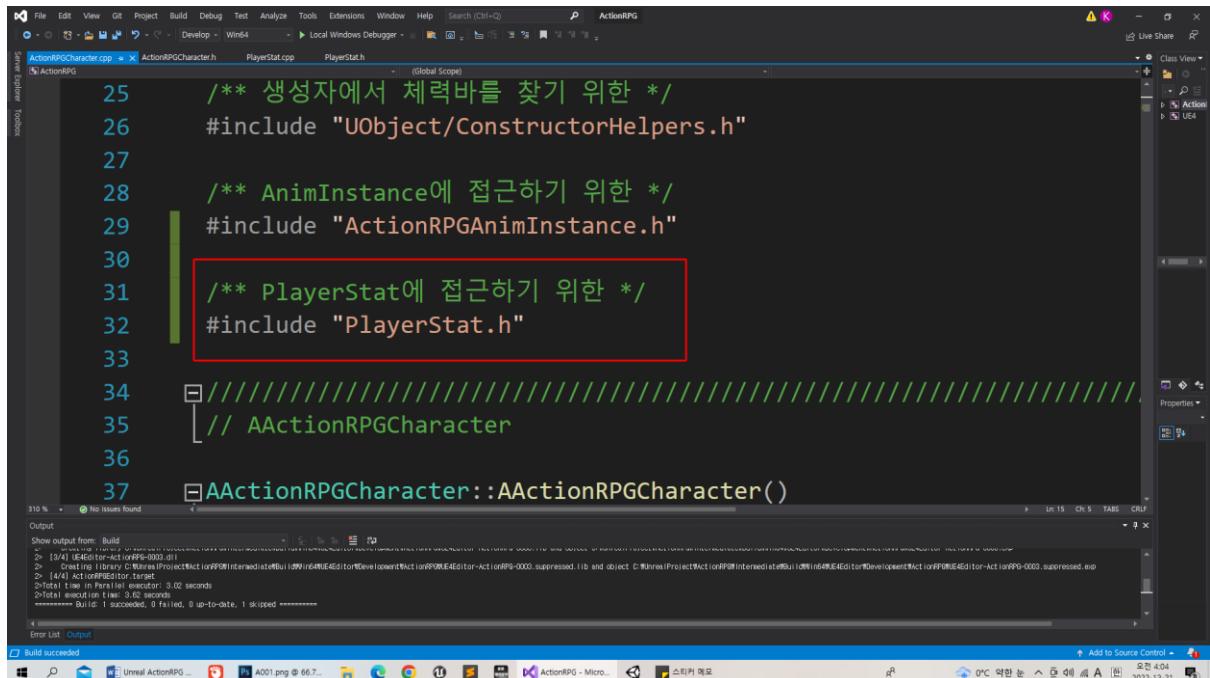
ActorComponent 클래스를 부모 클래스로 상속받는 PlayerStat이라는 이름의 클래스를 정의해 줍니다.





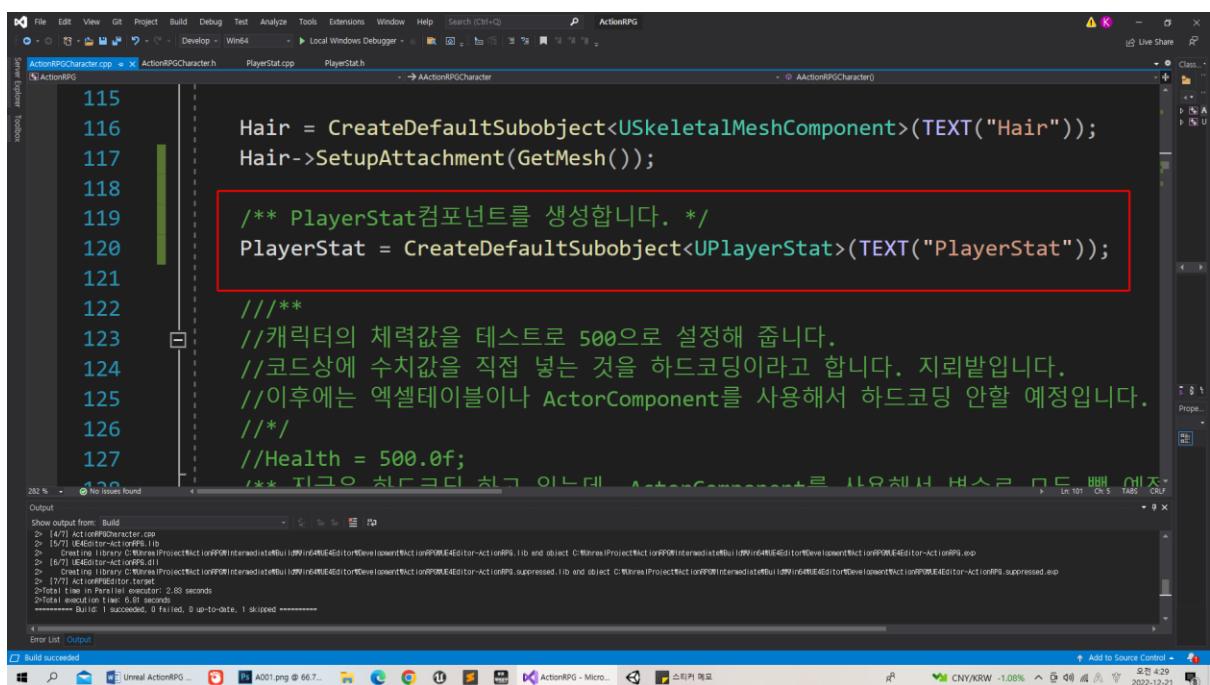
플레이어 클래스에서 생성해 주도록 합니다.





```
25     /** 생성자에서 체력바를 찾기 위한 */
26     #include "UObject/ConstructorHelpers.h"
27
28     /** AnimInstance에 접근하기 위한 */
29     #include "ActionRPGAnimInstance.h"
30
31     /** PlayerStat에 접근하기 위한 */
32     #include "PlayerStat.h"
33
34     //////////////////////////////////////////////////////////////////
35     // AActionRPGCharacter
36
37     AAActionRPGCharacter::AAActionRPGCharacter()
```

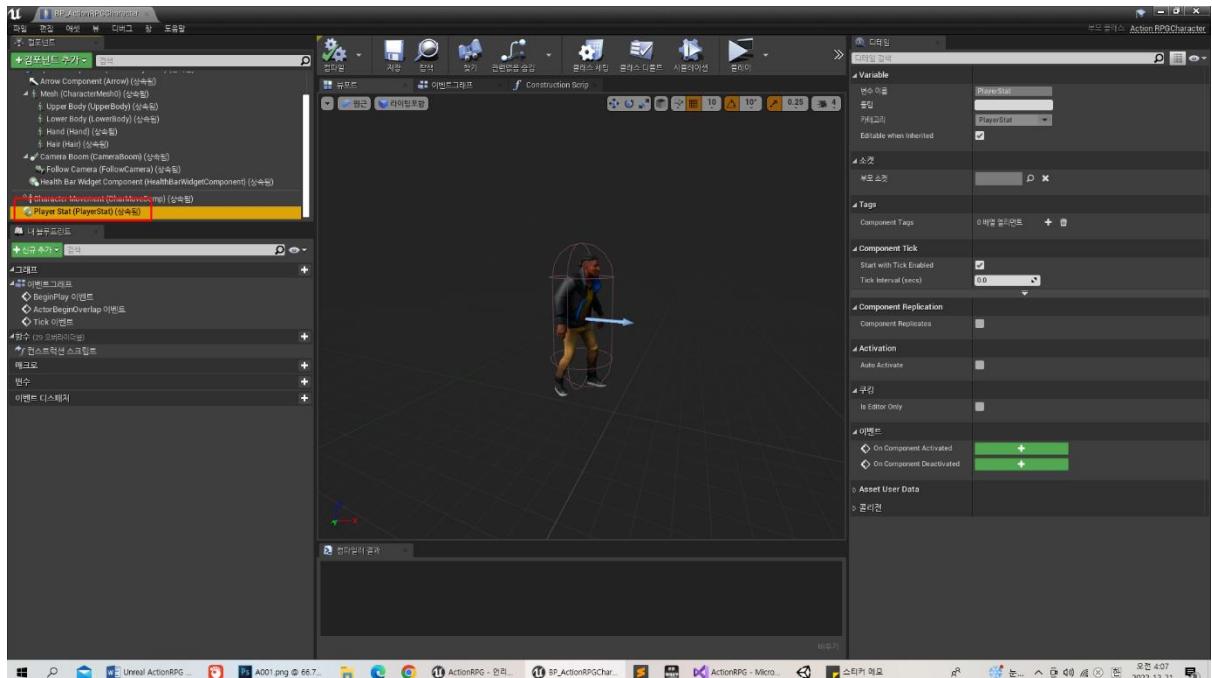
Output:  
Build succeeded  
2023-12-21 20:40:40



```
115
116
117     Hair = CreateDefaultSubobject<USkeletalMeshComponent>(TEXT("Hair"));
118     Hair->SetupAttachment(GetMesh());
119
120     /** PlayerStat컴포넌트를 생성합니다. */
121     PlayerStat = CreateDefaultSubobject<UPlayerStat>(TEXT("PlayerStat"));
122
123     /**
124     //캐릭터의 체력값을 테스트로 500으로 설정해 줍니다.
125     //코드상에 수치값을 직접 넣는 것을 하드코딩이라고 합니다. 지뢰밭입니다.
126     //이후에는 엑셀레이블이나 ActorComponent를 사용해서 하드코딩 안할 예정입니다.
127     //*/
128     //Health = 500.0f;
```

Output:  
Build succeeded  
2023-12-21 20:40:40

결과를 확인해 봅니다.

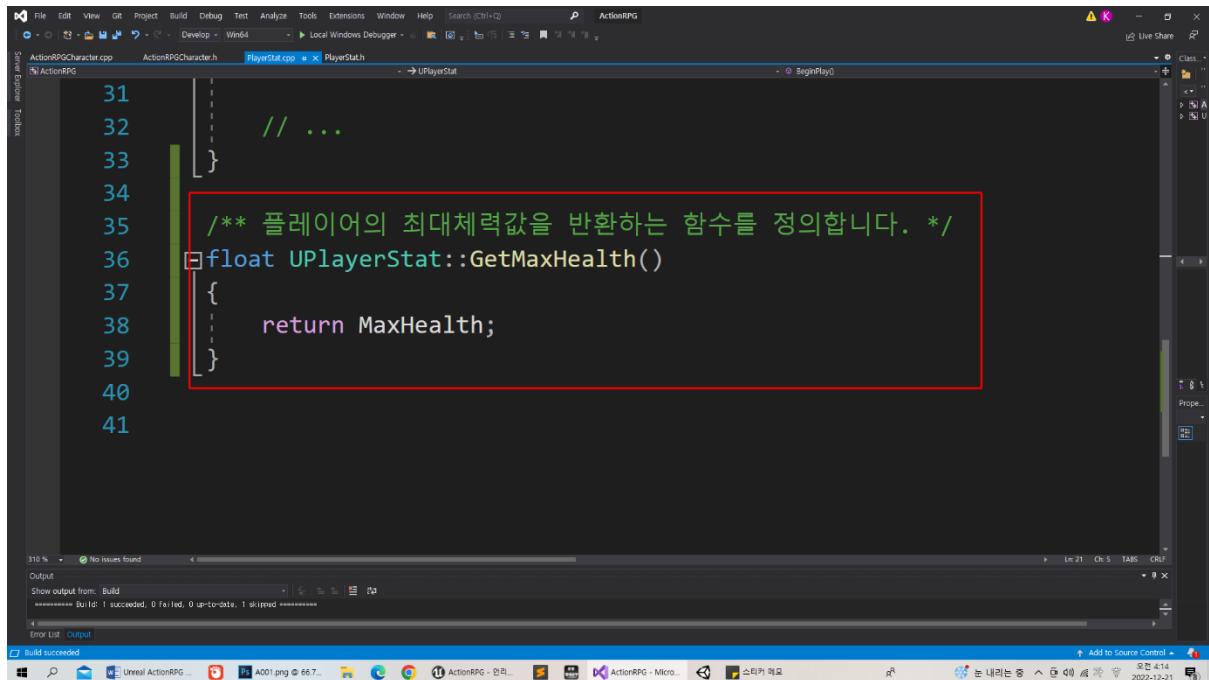


```

22
23     public:
24         // Called every frame
25         virtual void TickComponent(float DeltaTime, ELevelTick TickType, FActorComponentTickFunction* ThisTickFunction) override;
26
27     private:
28         /** 무기를 장착할 때 필요한 소켓이름은 저장할 변수를 선언합니다. */
29         UPROPERTY(EditAnywhere, Category = "PlayerStat", meta = (AllowPrivateAccess = "true"))
30         FName RightWeaponSocket;
31
32         /** 플레이어의 최대 체력값을 저장해 둘 변수를 선언합니다. */
33         UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = "Health", meta = (AllowPrivateAccess = "true"))
34         float MaxHealth;
35
36     public:
37         /** 플레이어의 최대체력값을 반환하는 함수의 원형을 선언합니다. */
38         float GetMaxHealth();
39     };
40

```

The code block shows the implementation of the PlayerStat component in C++. It includes private variables for weapon sockets and health, and a public function for getting the maximum health. The code is annotated with comments explaining the purpose of each section.

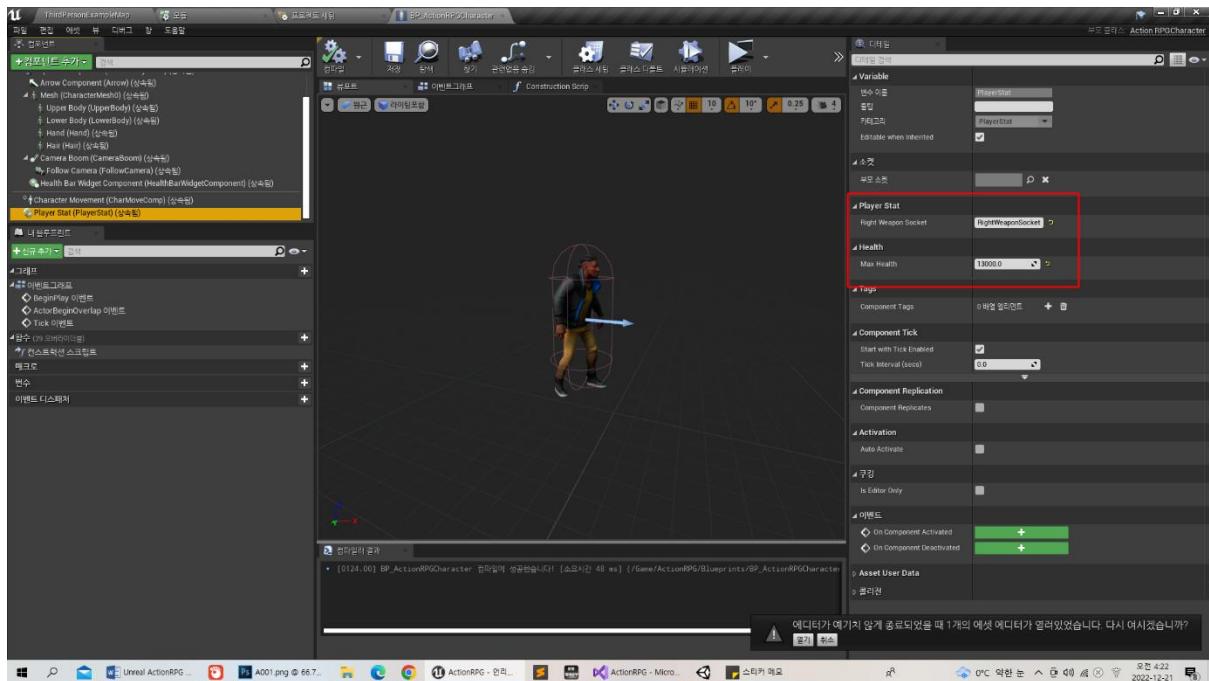


```

31 // ...
32 }
33
34
35 /** 플레이어의 최대체력값을 반환하는 함수를 정의합니다. */
36 float UPlayerStat::GetMaxHealth()
37 {
38     return MaxHealth;
39 }
40
41

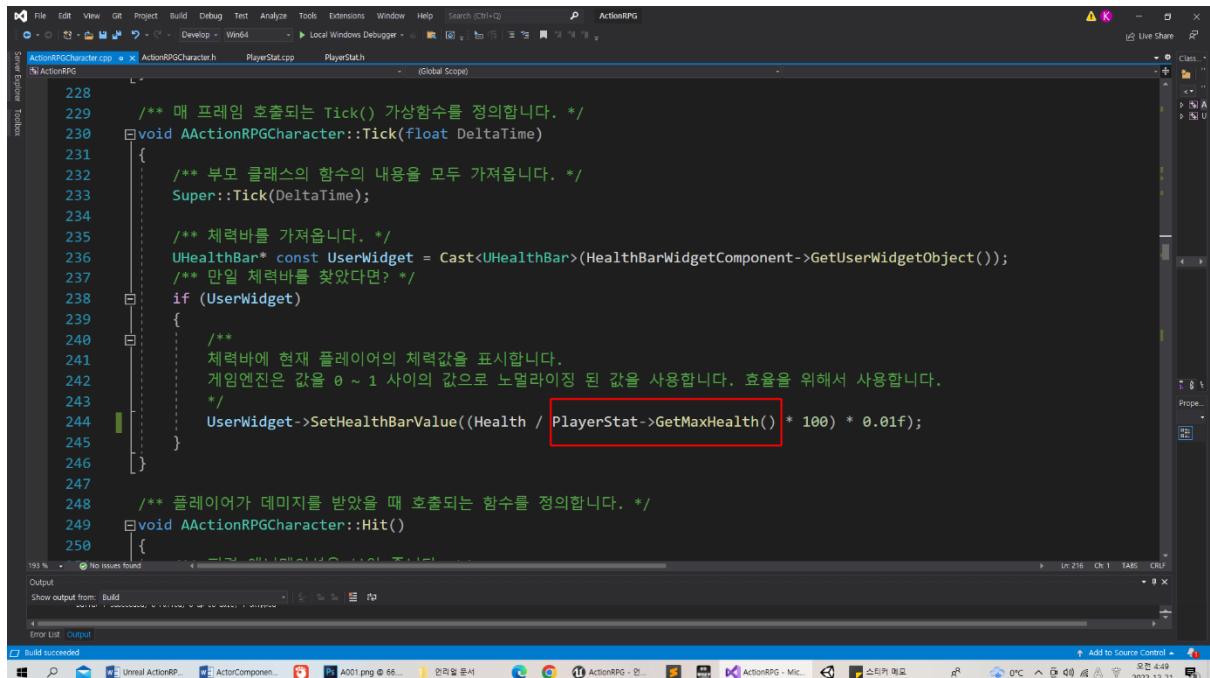
```

디테일 패널에서 확인해 봅니다.



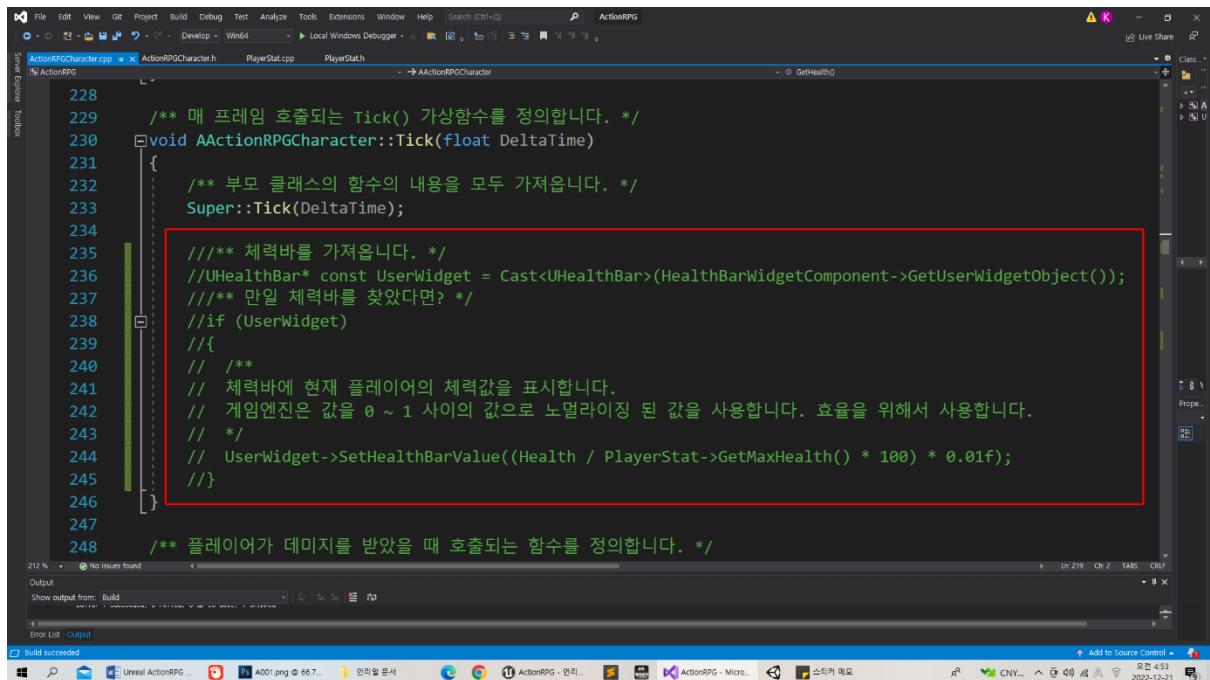
```
119  /** PlayerStat컴포넌트를 생성합니다. */
120  PlayerStat = CreateDefaultSubobject<UPlayerStat>(TEXT("PlayerStat"));
121  // PlayerStat->RegisterComponent();
122
123  /**
124  //캐릭터의 체력값을 테스트로 500으로 설정해 줍니다.
125  //코드상에 수치값을 직접 넣는 것을 하드코딩이라고 합니다. 지뢰발입니다.
126  //이후에는 엑셀테이블이나 ActorComponent를 사용해서 하드코딩 안할 예정입니다.
127  /**
128  //Health = 500.0f;
129  //** 지금은 하드코딩 하고 있는데, ActorComponent를 사용해서 변수로 모두 뺄 예정입니다. */
130  //** 플레이어의 최대 체력값*/
131  // MaxHealth = 13000.0f;
132  // Health = MaxHealth;
133
134  /**
135  //위젯 컴포넌트를 생성해 줍니다.
136  HealthBarWidgetComponent = CreateDefaultSubobject<UWidgetComponent>(TEXT("HealthBarWidgetComponent"));
137
138  /**
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
```

```
347  /**
348  // 부모클래스의 BeginPlay() 함수를 정의합니다.
349  void AACTIONRPGCharacter::BeginPlay()
350  {
351  // 부모 클래스의 함수의 내용을 모두 가져옵니다.
352  Super::BeginPlay();
353
354  // 플레이어의 무기들을 생성하고 기본 무기를 플레이어에게 장착합니다.
355  SpawnPlayerWeapon();
356
357  /**
358  //캐릭터의 체력값을 테스트로 500으로 설정해 줍니다.
359  //코드상에 수치값을 직접 넣는 것을 하드코딩이라고 합니다. 지뢰발입니다.
360  //이후에는 엑셀테이블이나 ActorComponent를 사용해서 하드코딩 안할 예정입니다.
361  /**
362  //Health = 500.0f;
363  //** 지금은 하드코딩 하고 있는데, ActorComponent를 사용해서 변수로 모두 뺄 예정입니다. */
364  //** 플레이어의 최대 체력값*/
365  // MaxHealth = 13000.0f;
366  // Health = MaxHealth;
367  Health = PlayerStat->GetMaxHealth();
368
369
```

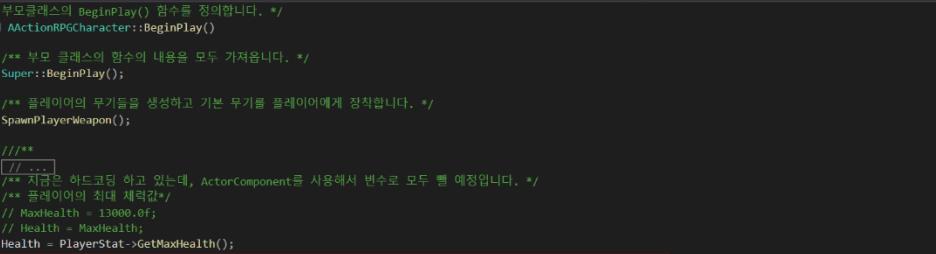


```
228  /** 매 프레임 호출되는 Tick() 가상함수를 정의합니다. */
229  void AActionRPGCharacter::Tick(float DeltaTime)
230  {
231      /** 부모 클래스의 함수의 내용을 모두 가져옵니다. */
232      Super::Tick(DeltaTime);
233
234      /** 체력바를 가져옵니다. */
235      UHealthBar* const UserWidget = Cast<UHealthBar>(HealthBarWidgetComponent->GetUserWidgetObject());
236      /** 만일 체력바를 찾았다면. */
237      if (UserWidget)
238      {
239          /**
240          체력바에 현재 플레이어의 체력값을 표시합니다.
241          게임엔진은 값을 0 ~ 1 사이의 값으로 노멀라이징 된 값을 사용합니다. 효율을 위해서 사용합니다.
242          */
243          UserWidget->SetHealthBarValue((Health / PlayerStat->GetMaxHealth() * 100) * 0.01f);
244      }
245  }
246
247
248  /** 플레이어가 데미지를 받았을 때 호출되는 함수를 정의합니다. */
249  void AActionRPGCharacter::Hit()
250  {
251  }
```

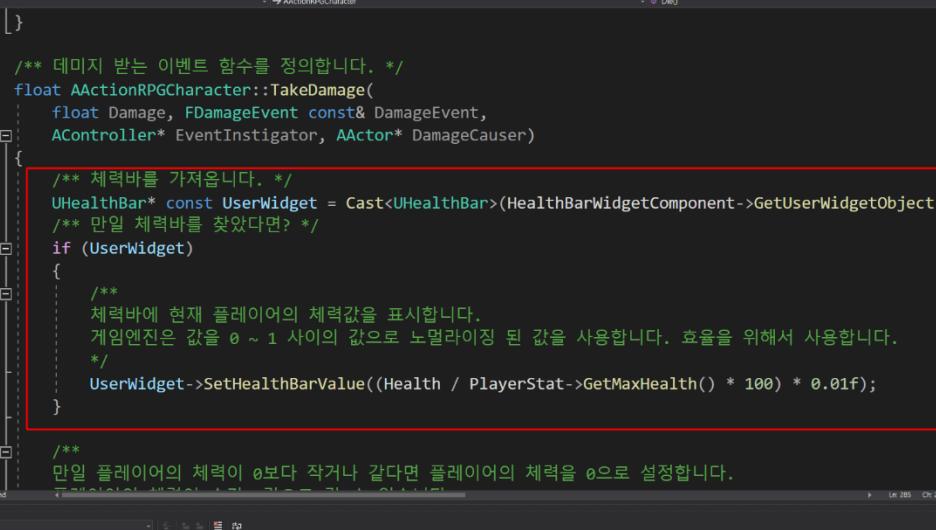
Tick() 함수 안에서 처리하면 부하가 많이 걸립니다. TakeDamage 함수안에서 처리해 주도록 합니다.



```
228  /** 매 프레임 호출되는 Tick() 가상함수를 정의합니다. */
229  void AActionRPGCharacter::Tick(float DeltaTime)
230  {
231      /** 부모 클래스의 함수의 내용을 모두 가져옵니다. */
232      Super::Tick(DeltaTime);
233
234      /**
235      // 체력바를 가져옵니다.
236      //UHealthBar* const UserWidget = Cast<UHealthBar>(HealthBarWidgetComponent->GetUserWidgetObject());
237      ///** 만일 체력바를 찾았다면. */
238      //if (UserWidget)
239      //{
240          /**
241          // 체력바에 현재 플레이어의 체력값을 표시합니다.
242          // 게임엔진은 값을 0 ~ 1 사이의 값으로 노멀라이징 된 값을 사용합니다. 효율을 위해서 사용합니다.
243          // */
244          // UserWidget->SetHealthBarValue((Health / PlayerStat->GetMaxHealth() * 100) * 0.01f);
245      //}
246  }
247
248  /** 플레이어가 데미지를 받았을 때 호출되는 함수를 정의합니다. */
249
```

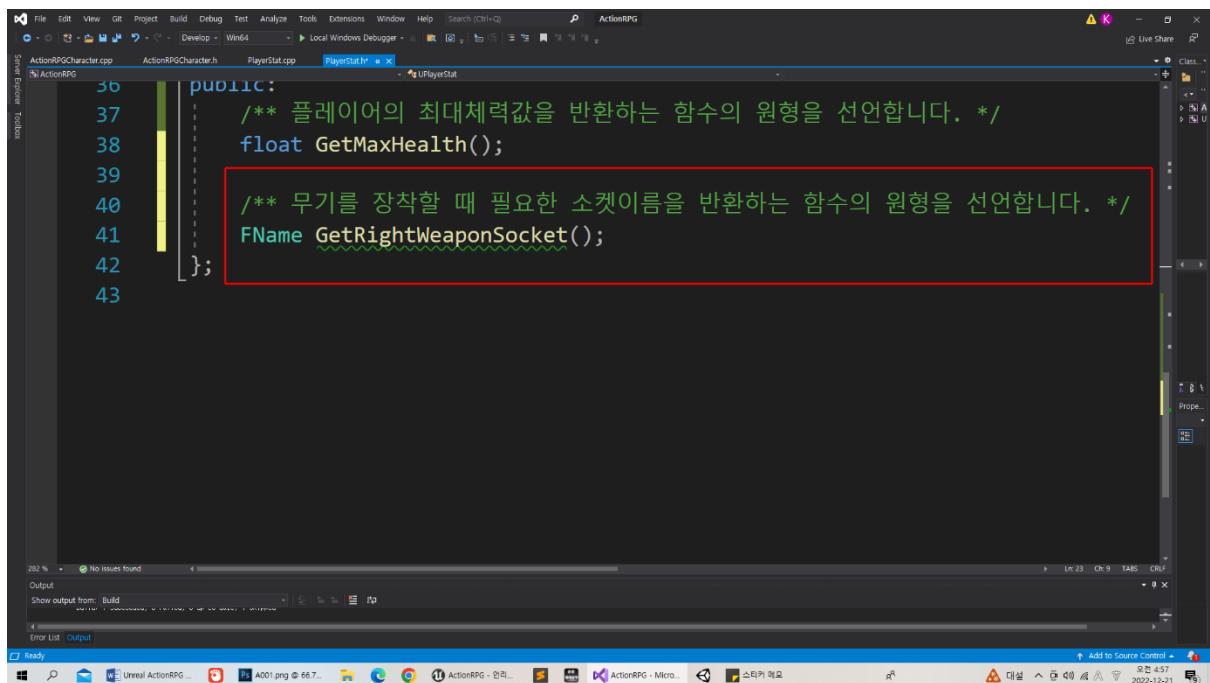


```
360     /** 부모클래스의 BeginPlay() 함수를 정의합니다. */
361     void AACTIONRPGCharacter::BeginPlay()
362     {
363         /** 부모 클래스의 함수의 내용을 모두 가져옵니다. */
364         Super::BeginPlay();
365
366         /** 플레이어의 무기들을 생성하고 기본 무기를 플레이어에게 장착합니다. */
367         SpawnPlayerWeapon();
368
369         //////
370         // ...
371         /** 지금은 하드코딩 하고 있는데, ActorComponent를 사용해서 변수로 모두 뺄 예정입니다. */
372         /** 플레이어의 최대 체력값*/
373         // MaxHealth = 13000.0f;
374         // Health = MaxHealth;
375         Health = PlayerStat->GetMaxHealth();
376
377         /** 체력바를 가져옵니다. */
378         UHealthBar* const UserWidget = Cast<UHealthBar>(HealthBarWidgetComponent-> GetUserWidgetObject());
379         /** 만일 체력바를 찾았다면? */
380         if (UserWidget)
381         {
382             /**
383             체력바에 현재 플레이어의 체력값을 표시합니다.
384             게임엔진은 값은 0 ~ 1 사이의 값으로 노말라이징 된 값을 사용합니다. 효율을 위해서 사용합니다.
385             */
386             UserWidget->SetHealthBarValue((Health / PlayerStat->GetMaxHealth() * 100) * 0.01f);
387         }
388
389         /** 사용자가 지정한 무기를 장착하는 함수를 정의합니다. */
390     }
391
392     // ...
393
394     // ...
395 }
```

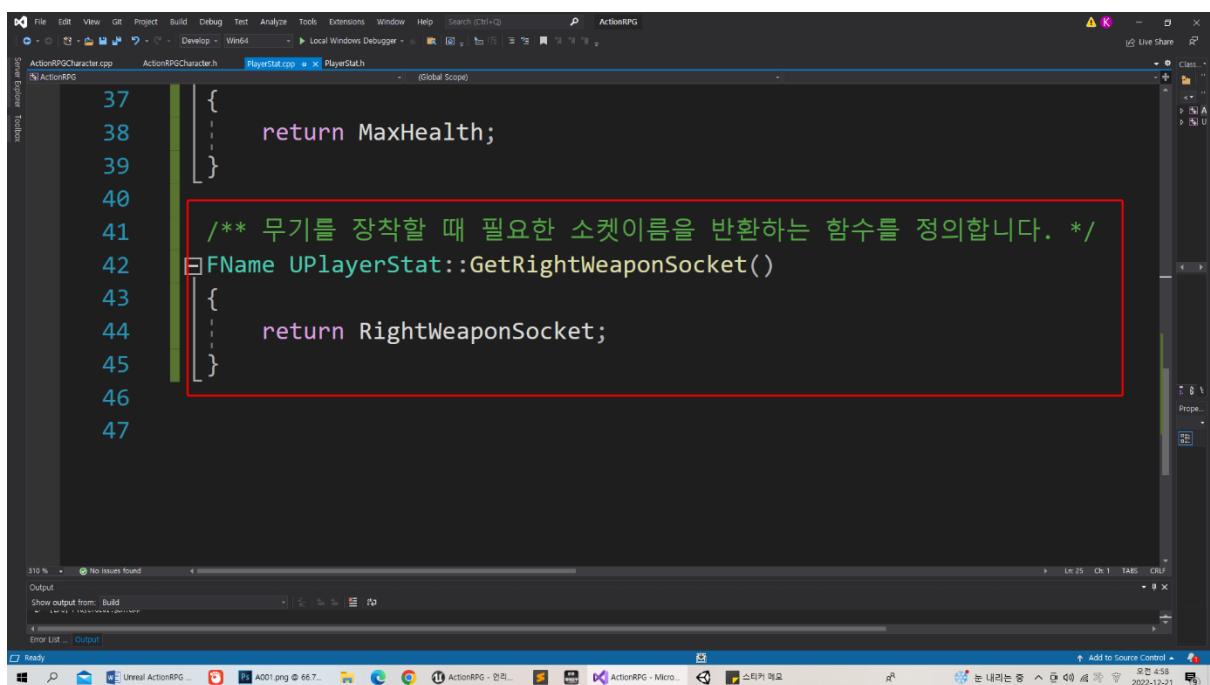


```
296     }
297
298     /** 데미지 받는 이벤트 함수를 정의합니다. */
299     float ActionRPGCharacter::TakeDamage(
300         float Damage, FDamageEvent const& DamageEvent,
301         AController* EventInstigator, AActor* DamageCauser)
302     {
303         /** 체력바를 가져옵니다. */
304         UHealthBar* const UserWidget = Cast<UHealthBar>(HealthBarWidgetComponent->GetUserWidgetObject());
305         /** 만일 체력바를 찾았다면? */
306         if (UserWidget)
307         {
308             /**
309             체력바에 현재 플레이어의 체력값을 표시합니다.
310             게임엔진은 값을 0 ~ 1 사이의 값으로 노멀라이징 된 값을 사용합니다. 효율을 위해서 사용합니다.
311             */
312             UserWidget->SetHealthBarValue((Health / PlayerStat->GetMaxHealth() * 100) * 0.01f);
313         }
314
315         /**
316         만일 플레이어의 체력이 0보다 작거나 같다면 플레이어의 체력을 0으로 설정합니다.
317         */
318     }
319 }
```

소켓을 반환하는 함수의 원형을 선언합니다.

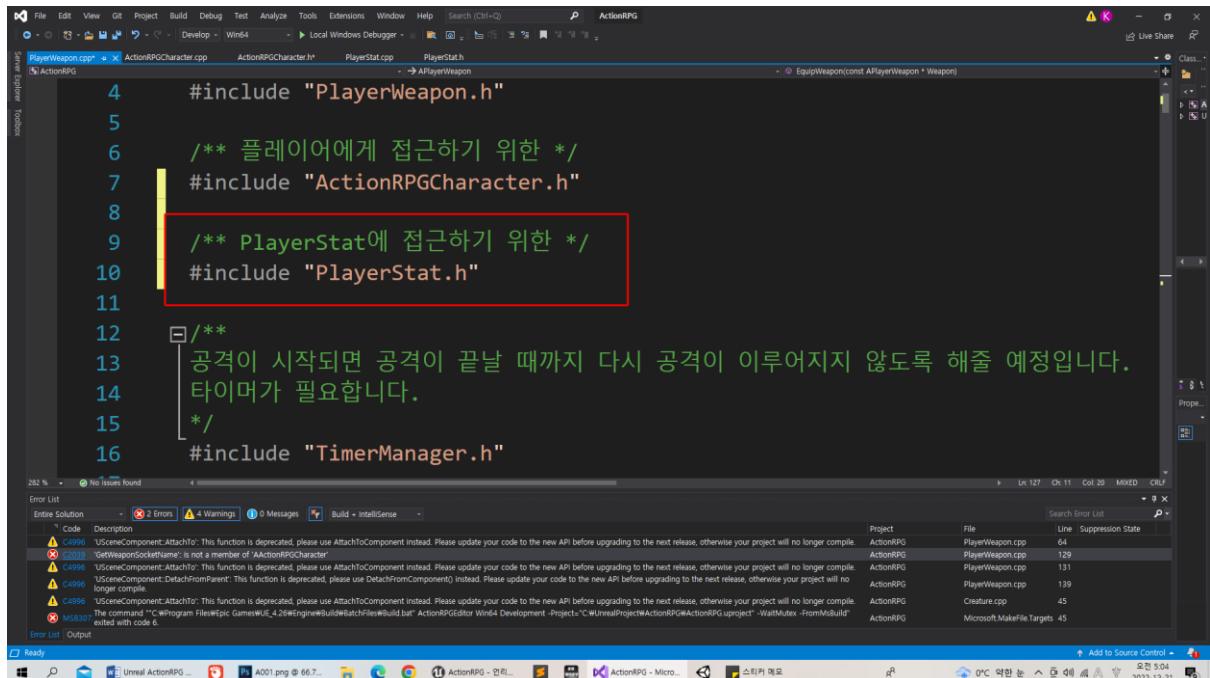


```
36     public:
37     /** 플레이어의 최대체력값을 반환하는 함수의 원형을 선언합니다. */
38     float GetMaxHealth();
39
40     /** 무기를 장착할 때 필요한 소켓이름을 반환하는 함수의 원형을 선언합니다. */
41     FName GetRightWeaponSocket();
42
43 }
```



```
37     {
38     return MaxHealth;
39     }
40
41     /** 무기를 장착할 때 필요한 소켓이름을 반환하는 함수를 정의합니다. */
42     FName UPlayerStat::GetRightWeaponSocket()
43     {
44     return RightWeaponSocket;
45     }
46
47 }
```

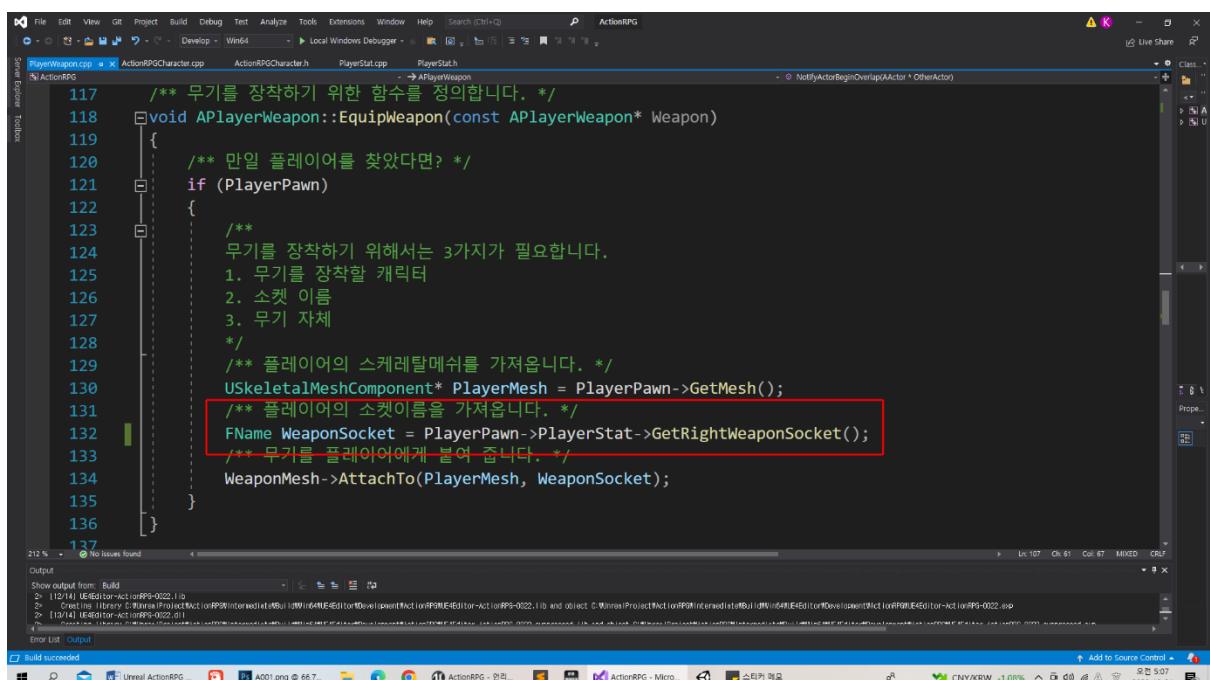
수정해 주도록 합니다.



```

4 #include "PlayerWeapon.h"
5
6 /* 플레이어에게 접근하기 위한 */
7 #include "ActionRPGCharacter.h"
8
9 /* PlayerStat에 접근하기 위한 */
10 #include "PlayerStat.h"
11
12 /**
13  * 공격이 시작되면 공격이 끝날 때까지 다시 공격이 이루어지지 않도록 해줄 예정입니다.
14  * 타이머가 필요합니다.
15 */
16 #include "TimerManager.h"

```

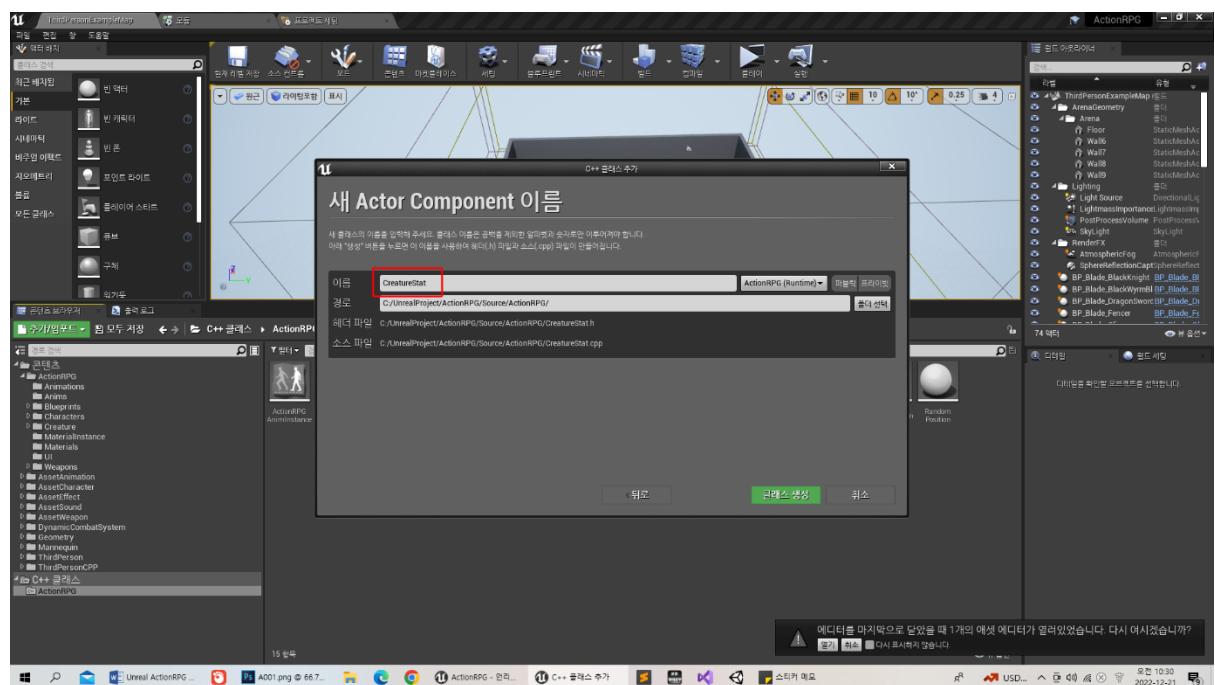
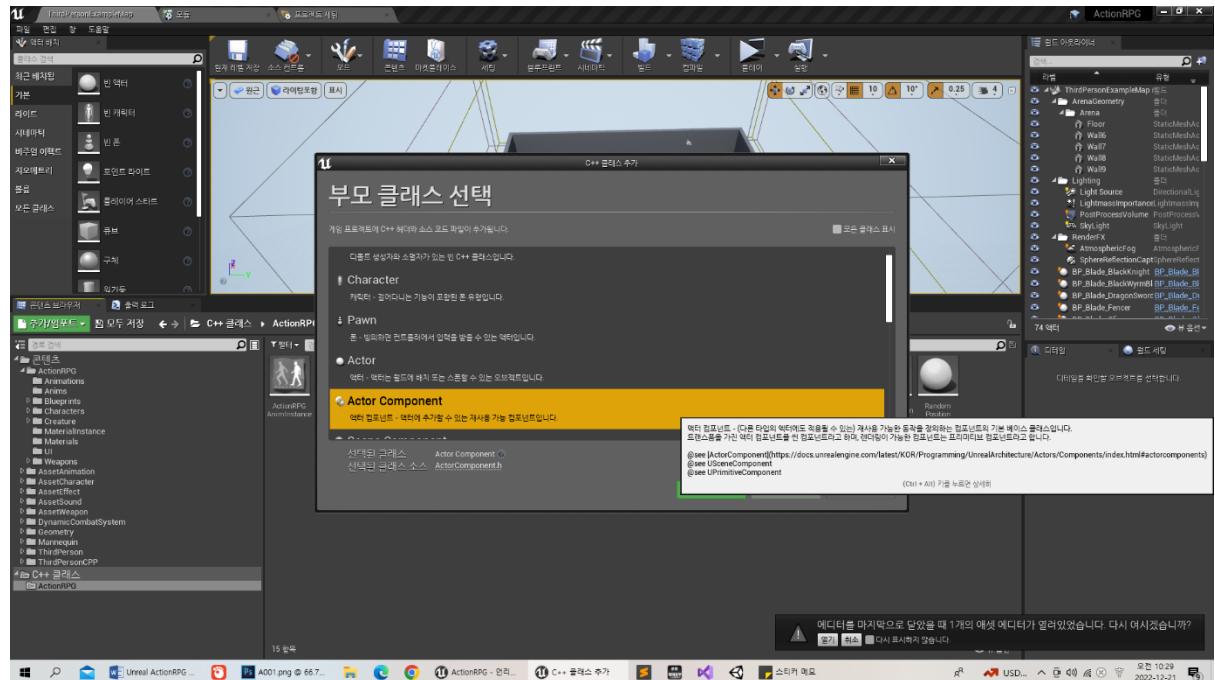


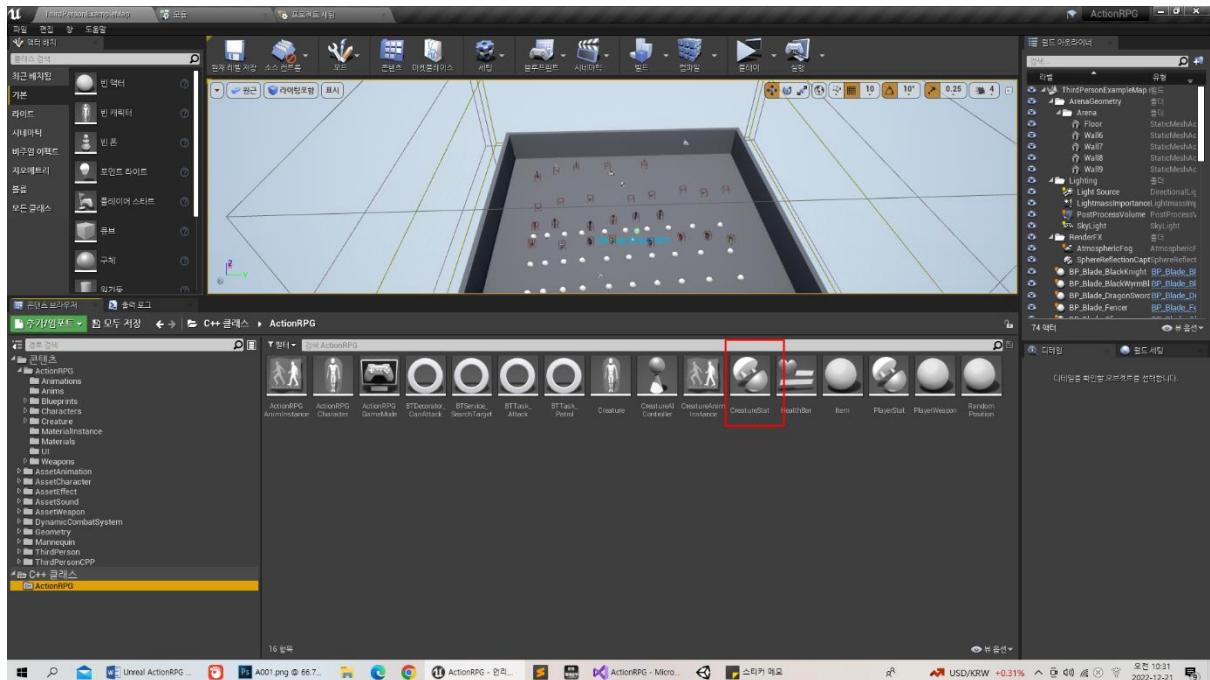
```

117 /**
118  * 무기를 장착하기 위한 함수를 정의합니다. */
119 void APlayerWeapon::EquipWeapon(const APlayerWeapon* Weapon)
120 {
121     /**
122     * 만일 플레이어를 찾았다면? */
123     if (PlayerPawn)
124     {
125         /**
126         * 무기를 장착하기 위해서는 3가지가 필요합니다.
127         1. 무기를 장착할 캐릭터
128         2. 소켓 이름
129         3. 무기 자체
130         */
131         USkeletalMeshComponent* PlayerMesh = PlayerPawn->GetMesh();
132         /**
133         * 플레이어의 소켓이름을 가져옵니다. */
134         FName WeaponSocket = PlayerStat->GetRightWeaponSocket();
135         /**
136         * 무기를 플레이어에게 블어 줍니다. */
137         WeaponMesh->AttachTo(PlayerMesh, WeaponSocket);
138     }
139 }

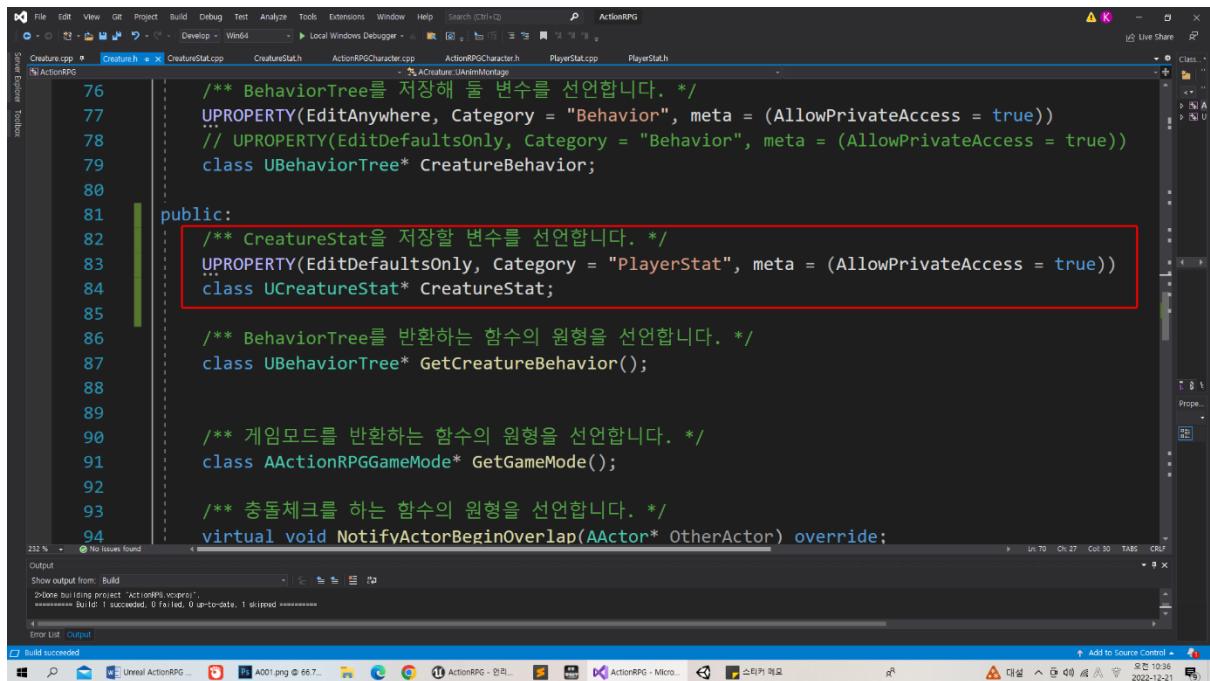
```

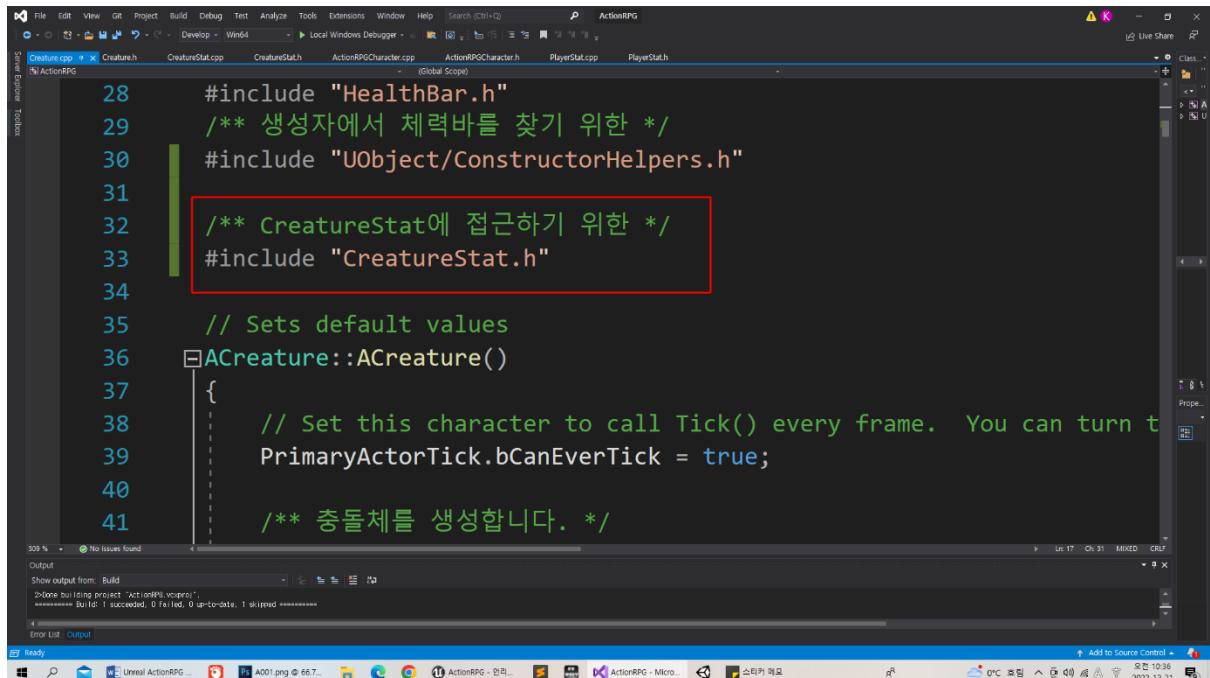
이제 크리쳐쪽도 해 주도록 합니다. ,ActorComponent를 부모 클래스로 선택하는 CreatureStat이라는 클래스를 정의해 주도록 합니다.



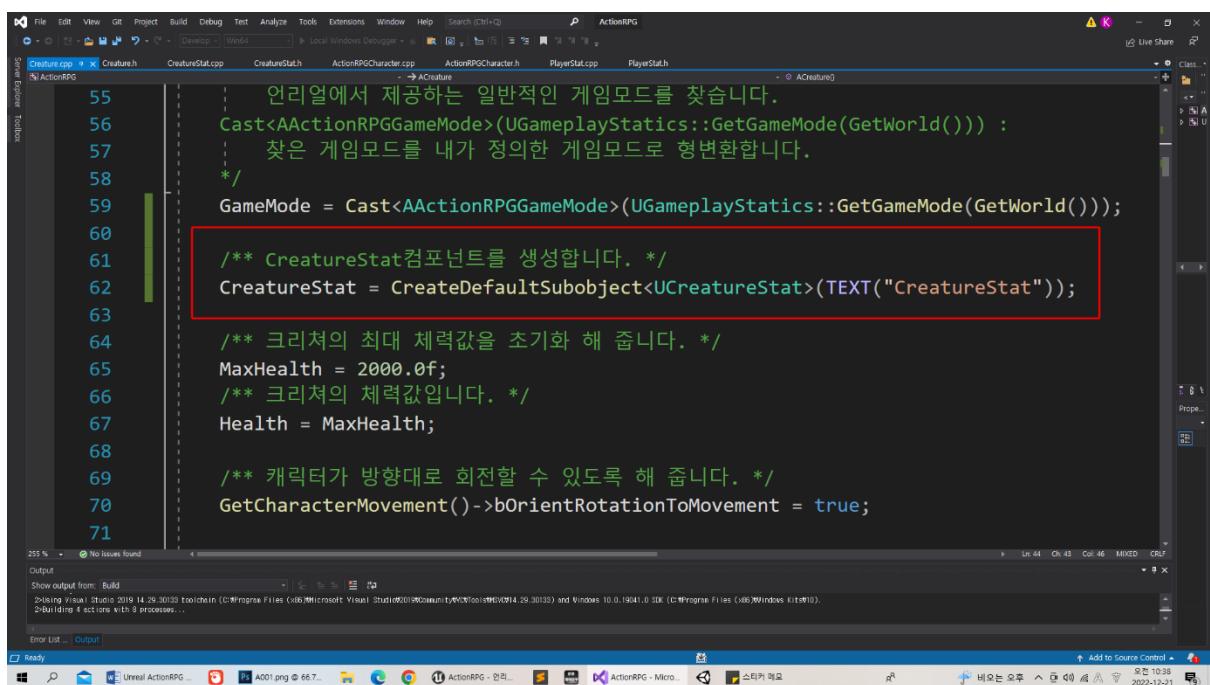


생성자 함수에서 생성해 주도록 합니다.



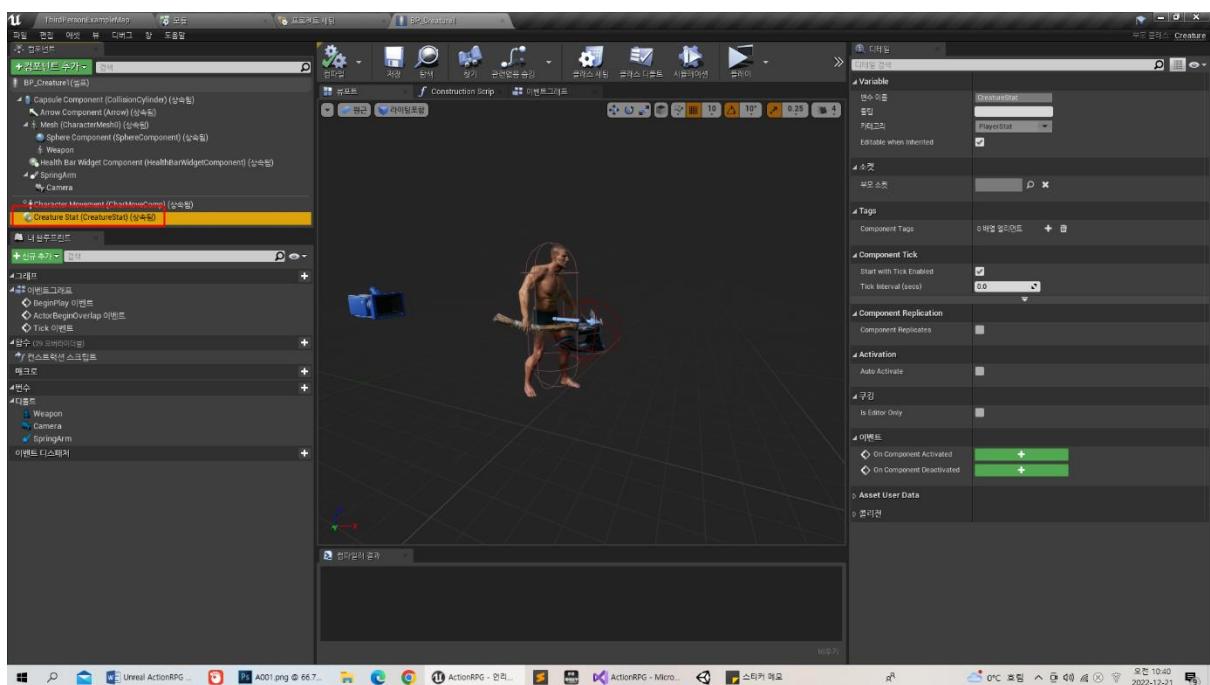
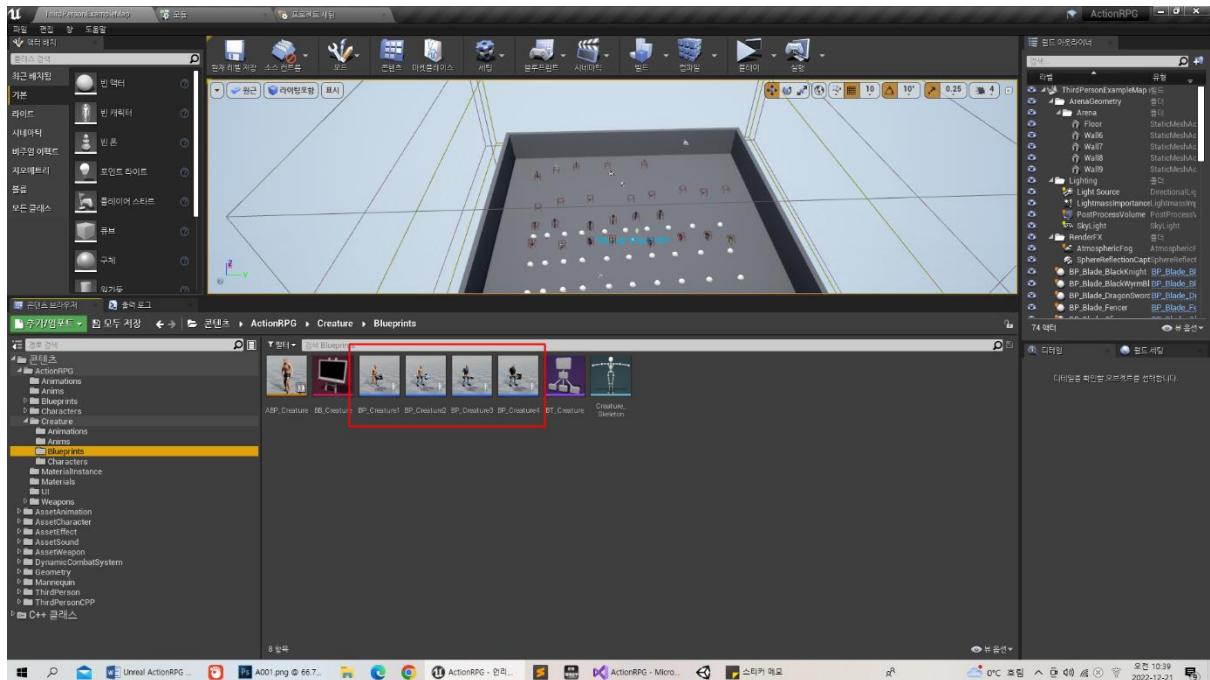


```
28     #include "HealthBar.h"
29     /** 생성자에서 체력바를 찾기 위한 */
30     #include "UObject/ConstructorHelpers.h"
31
32     /** CreatureStat에 접근하기 위한 */
33     #include "CreatureStat.h"
34
35     // Sets default values
36     ACreature::ACreature()
37     {
38         // Set this character to call Tick() every frame. You can turn this off in the Settings panel
39         PrimaryActorTick.bCanEverTick = true;
40
41         /** 충돌체를 생성합니다. */
```

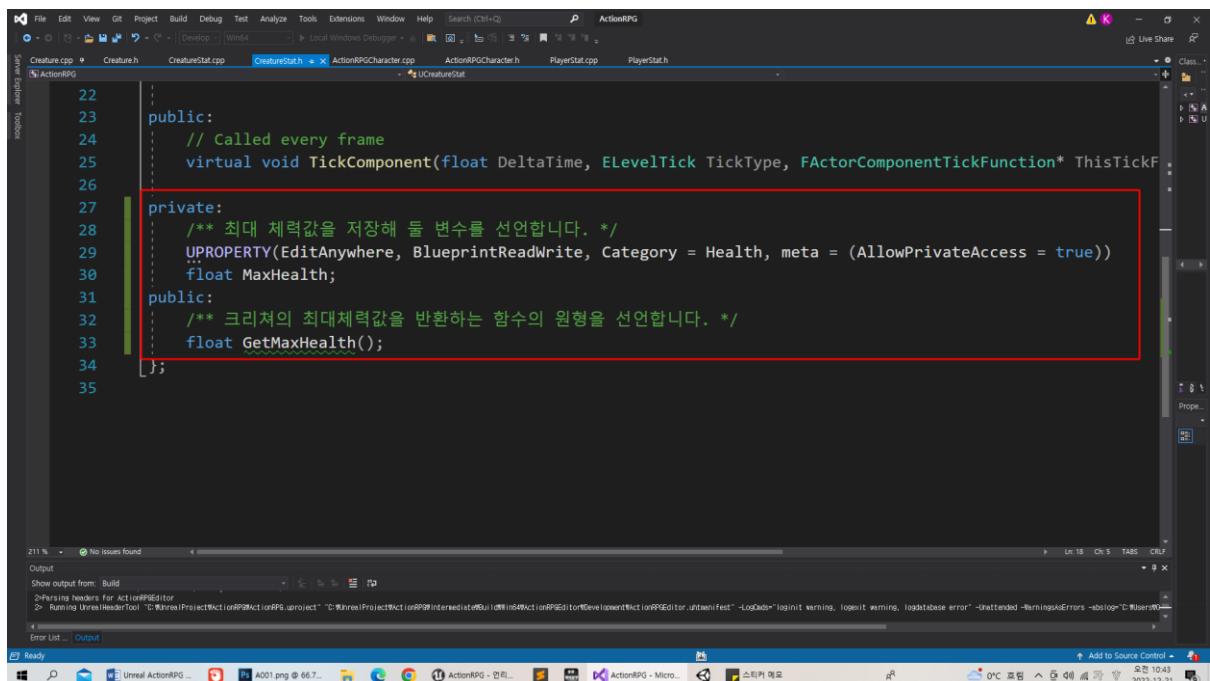


```
55     언리얼에서 제공하는 일반적인 게임모드를 찾습니다.
56     Cast<AACTIONRGGameMode>(UGameplayStatics::GetGameMode(GetWorld())) :
57         찾은 게임모드를 내가 정의한 게임모드로 형변환합니다.
58     */
59     GameMode = Cast<AACTIONRGGameMode>(UGameplayStatics::GetGameMode(GetWorld()));
60
61     /** CreatureStat컴포넌트를 생성합니다. */
62     CreatureStat = CreateDefaultSubobject<UCreatureStat>(TEXT("CreatureStat"));
63
64     /** 크리쳐의 최대 체력값을 초기화 해 줍니다. */
65     MaxHealth = 2000.0f;
66     /** 크리쳐의 체력값입니다. */
67     Health = MaxHealth;
68
69     /** 캐릭터가 방향대로 회전할 수 있도록 해 줍니다. */
70     GetCharacterMovement()->bOrientRotationToMovement = true;
71
```

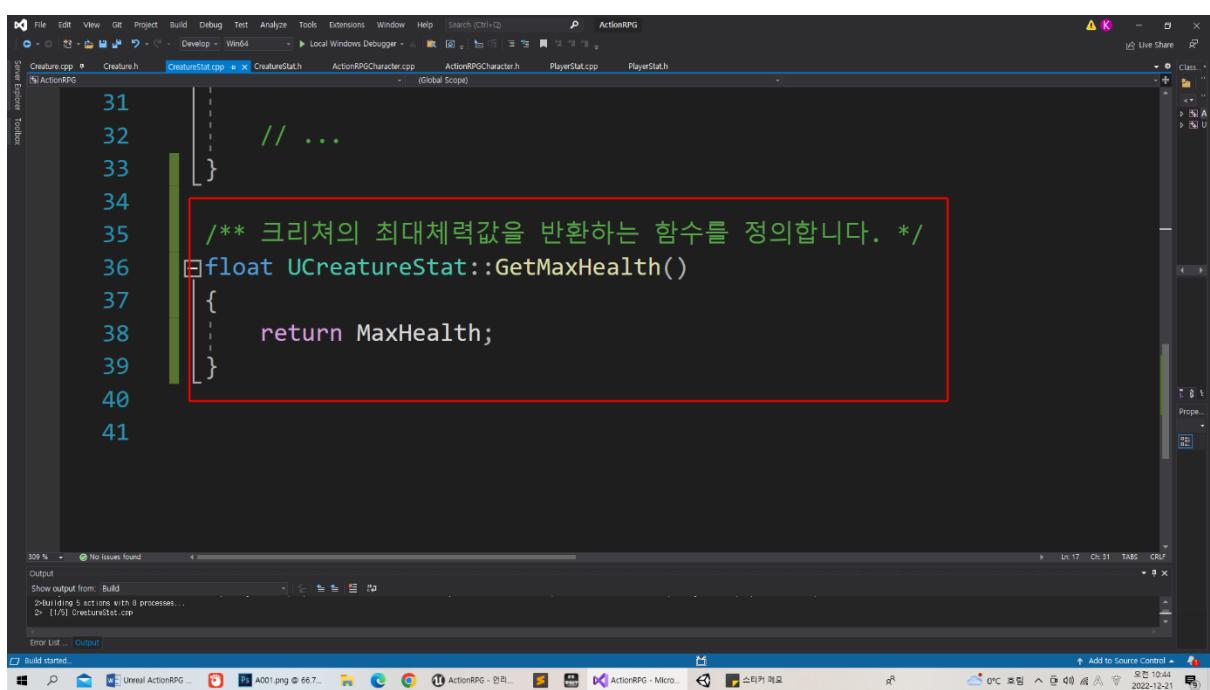
결과를 확인합니다.



크리쳐의 체력부터 적용해 주도록 합니다.

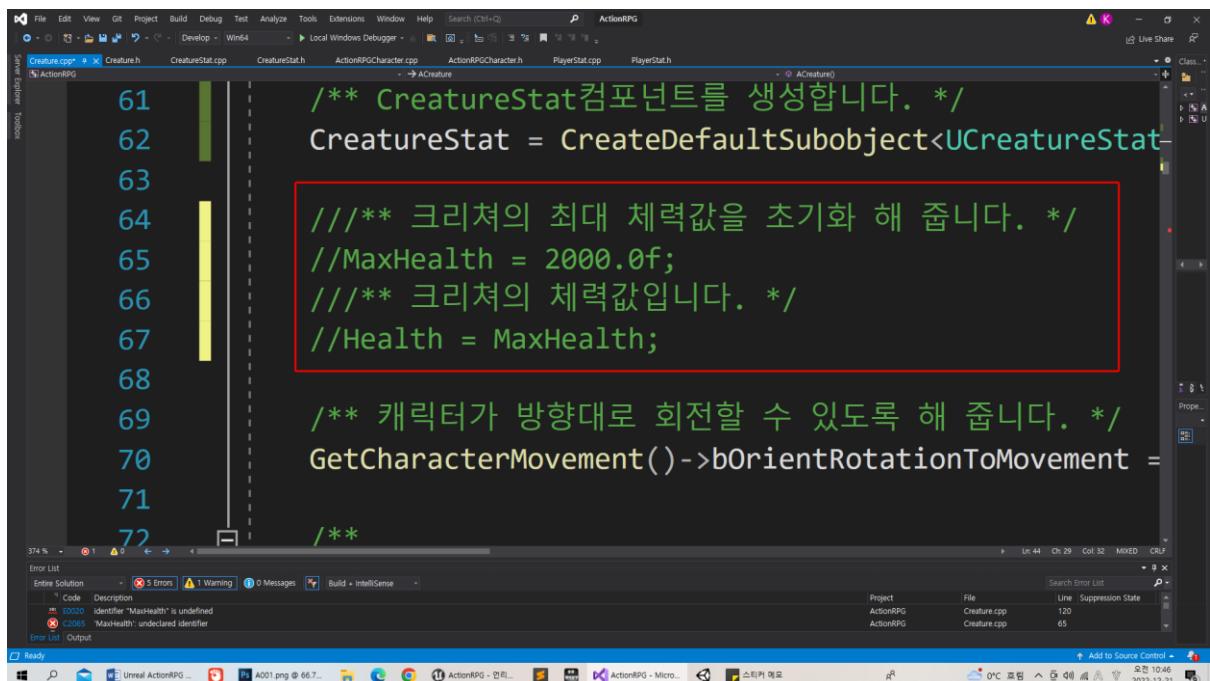


```
22
23     public:
24         // Called every frame
25         virtual void TickComponent(float DeltaTime, ELevelTick TickType, FActorComponentTickFunction* ThisTickFunction) override;
26
27     private:
28         /** 최대 체력값을 저장해 둘 변수를 선언합니다. */
29         UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = Health, meta = (AllowPrivateAccess = true))
30         float MaxHealth;
31
32     public:
33         /** 크리쳐의 최대체력값을 반환하는 함수의 원형을 선언합니다. */
34         float GetMaxHealth();
35     
```



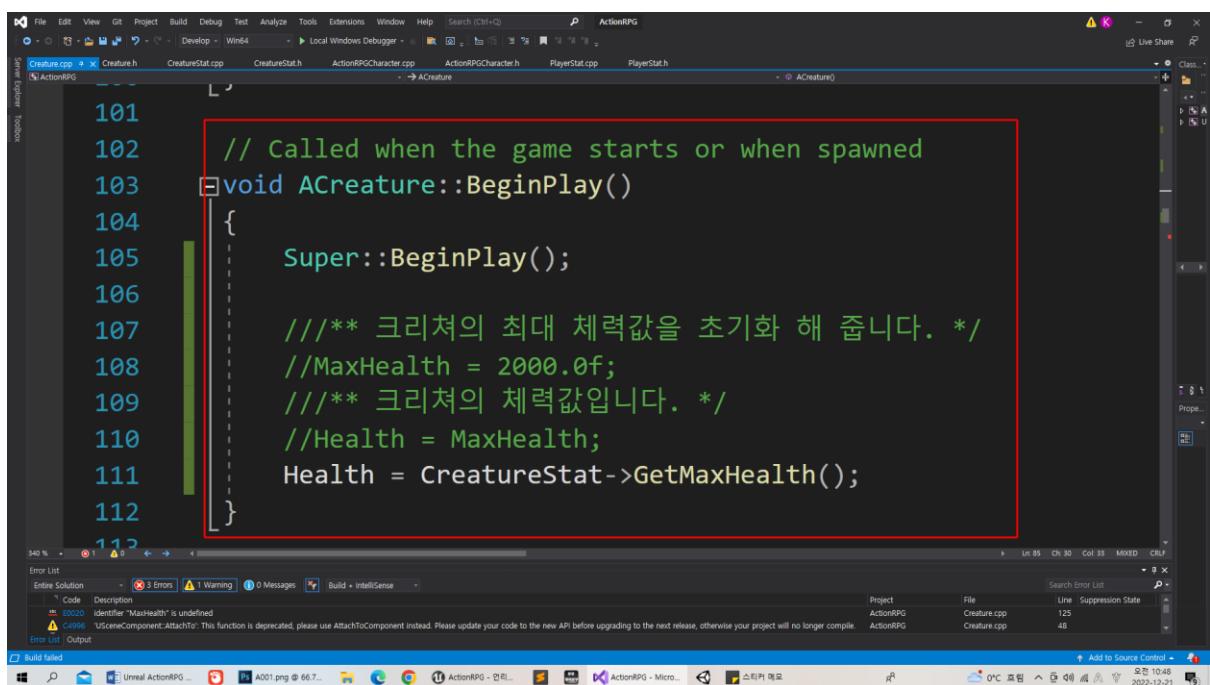
```
31
32     // ...
33
34
35     /** 크리쳐의 최대체력값을 반환하는 함수를 정의합니다. */
36     float UCreatureStat::GetMaxHealth()
37     {
38         return MaxHealth;
39     }
40
41

```

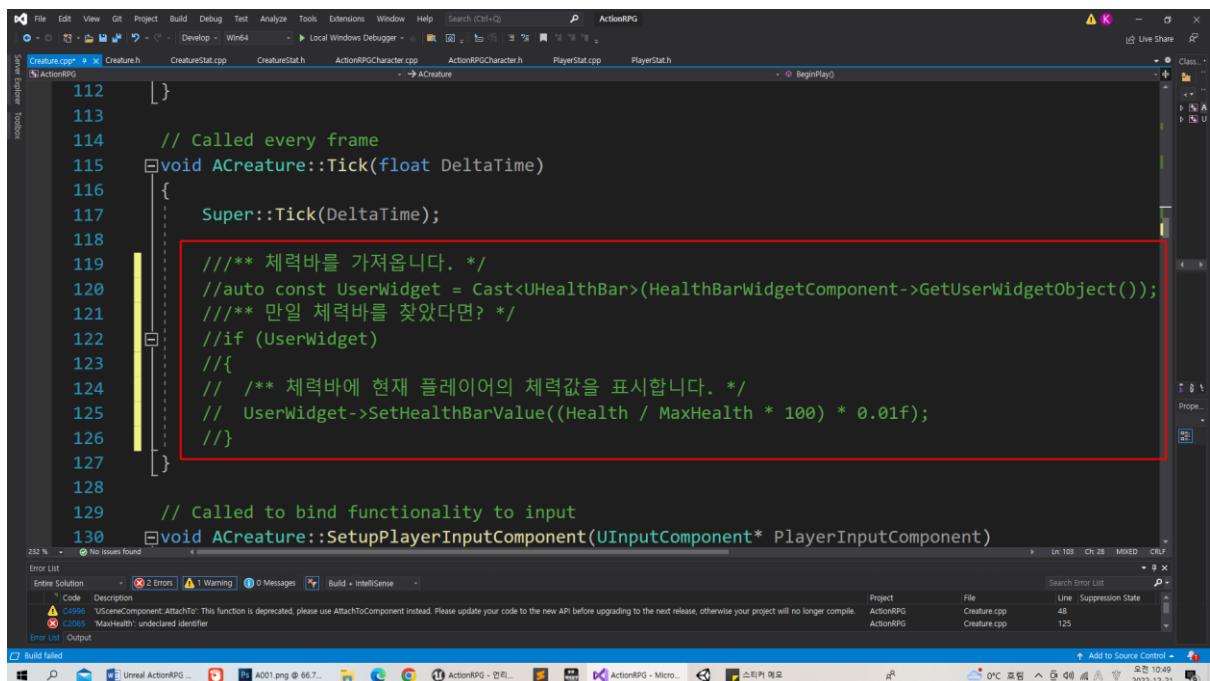


```
61  /** CreatureStat컴포넌트를 생성합니다. */
62  CreatureStat = CreateDefaultSubobject<UCreatureStat>(GetTransientByName("CreatureStat"));
63
64  // /** 크리쳐의 최대 체력값을 초기화 해 줍니다. */
65  // MaxHealth = 2000.0f;
66  // /** 크리쳐의 체력값입니다. */
67  // Health = MaxHealth;
68
69  /** 캐릭터가 방향대로 회전할 수 있도록 해 줍니다. */
70  GetCharacterMovement()->bOrientRotationToMovement = true;
71
72  /**
73
74
75
76
77
78
79
79
80
81
82
83
84
85
86
87
88
89
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
```

101 // Called when the game starts or when spawned  
102 void ACreature::BeginPlay()  
103 {  
104 Super::BeginPlay();  
105  
106 // /\*\* 크리쳐의 최대 체력값을 초기화 해 줍니다. \*/  
107 // MaxHealth = 2000.0f;  
108 // /\*\* 크리쳐의 체력값입니다. \*/  
109 // Health = MaxHealth;  
110 Health = CreatureStat->GetMaxHealth();  
111 }  
112 }



```
101 // Called when the game starts or when spawned
102 void ACreature::BeginPlay()
103 {
104     Super::BeginPlay();
105
106     // /** 크리쳐의 최대 체력값을 초기화 해 줍니다. */
107     // MaxHealth = 2000.0f;
108     // /** 크리쳐의 체력값입니다. */
109     // Health = MaxHealth;
110     Health = CreatureStat->GetMaxHealth();
111 }
```



```
112     }
113
114     // Called every frame
115     void ACreature::Tick(float DeltaTime)
116     {
117         Super::Tick(DeltaTime);
118
119         /**
120          * 체력바를 가져옵니다.
121          */
122         auto const UserWidget = Cast<UHealthBar>(HealthBarWidgetComponent->GetUserWidgetObject());
123
124         /**
125          * 만일 체력바를 찾았다면?
126          */
127         if (UserWidget)
128         {
129             /**
130              * 체력바에 현재 플레이어의 체력값을 표시합니다.
131              */
132             UserWidget->SetHealthBarValue((Health / MaxHealth * 100) * 0.01f);
133         }
134     }
135
136     // Called to bind functionality to input
137     void ACreature::SetupPlayerInputComponent(UInputComponent* PlayerInputComponent)
```

232 % No issues found

Error List

Entire Solution - 2 Errors, 1 Warning, 0 Messages, Build + IntelliSense

UE4-2095 'USceneComponent::AttachTo': This function is deprecated, please use AttachToComponent instead. Please update your code to the new API before upgrading to the next release, otherwise your project will no longer compile.

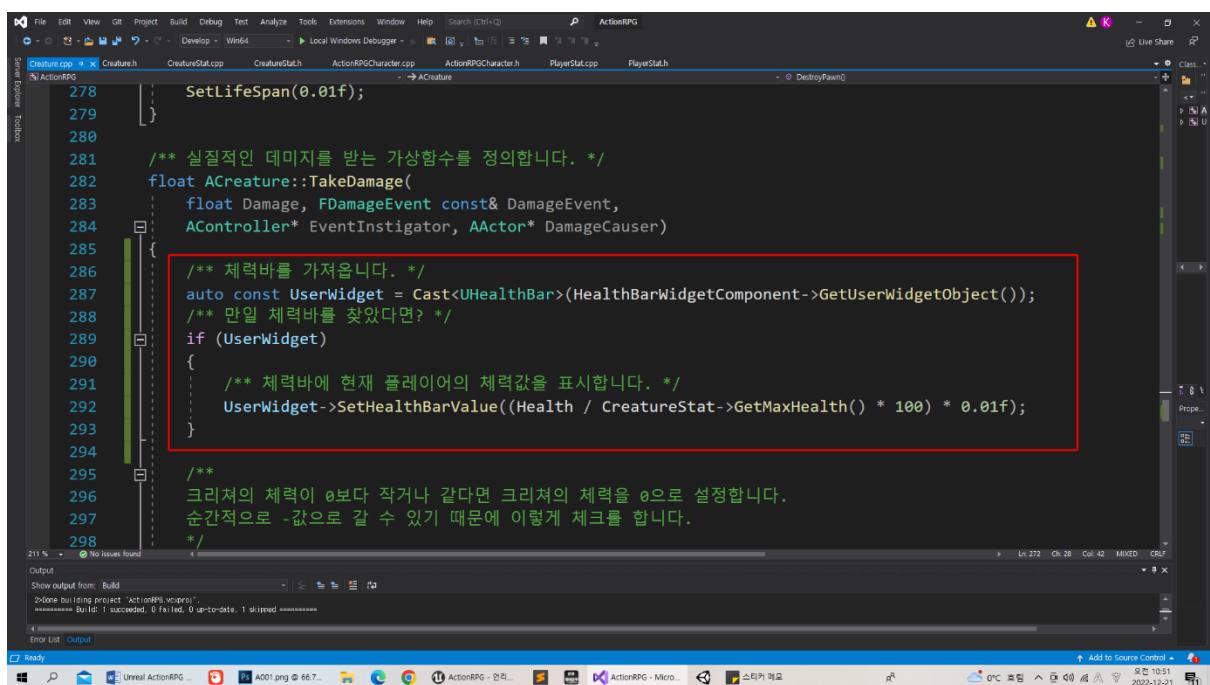
UE4-2095 'MaxHealth': undeclared identifier

Output

Build failed

Unreal ActionRPG... A001.png @ 66.7... ActionRPG - 언리... ActionRPG - Micro... 스티커 메모

0xC 흐름 10:49 2022-12-21



```
278     SetLifeSpan(0.01f);
279 }
280
281 /**
282  * 실질적인 데미지를 받는 가상함수를 정의합니다.
283  */
284 float ACreature::TakeDamage(
285     float Damage, FDamageEvent const& DamageEvent,
286     AController* EventInstigator, AActor* DamageCauser)
287 {
288
289     /**
290      * 체력바를 가져옵니다.
291      */
292     auto const UserWidget = Cast<UHealthBar>(HealthBarWidgetComponent->GetUserWidgetObject());
293
294     /**
295      * 만일 체력바를 찾았다면?
296      */
297     if (UserWidget)
298     {
299
300         /**
301          * 체력바에 현재 플레이어의 체력값을 표시합니다.
302          */
303         UserWidget->SetHealthBarValue((Health / CreatureStat->GetMaxHealth() * 100) * 0.01f);
304     }
305
306     /**
307      * 크리쳐의 체력이 0보다 작거나 같다면 크리쳐의 체력을 0으로 설정합니다.
308      * 순간적으로 -값으로 갈 수 있기 때문에 이렇게 체크를 합니다.
309      */
310 }
```

211 % No issues found

Error List

Entire Solution - 2 Errors, 1 Warning, 0 Messages, Build + IntelliSense

UE4-2095 'USceneComponent::AttachTo': This function is deprecated, please use AttachToComponent instead. Please update your code to the new API before upgrading to the next release, otherwise your project will no longer compile.

UE4-2095 'MaxHealth': undeclared identifier

Output

Show output from: Build

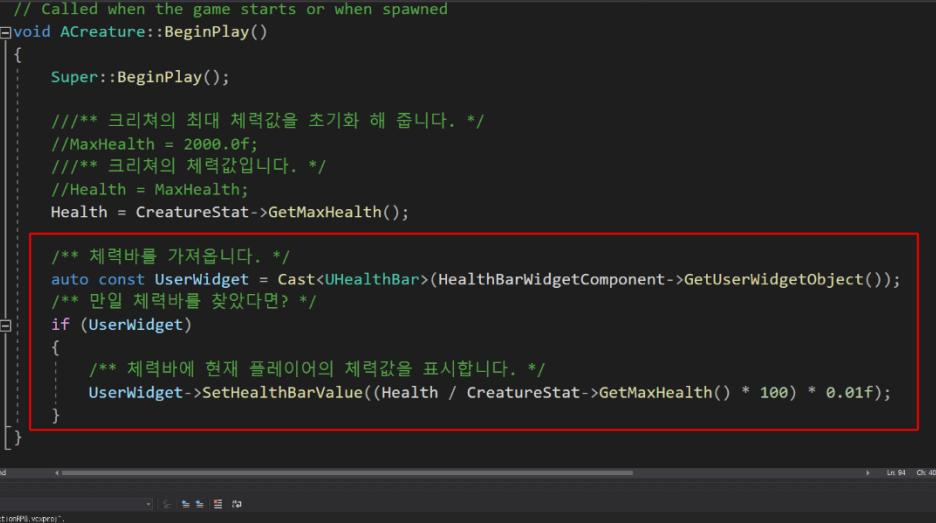
2022-12-21 10:51

Build 1 succeeded, 0 failed, 0 up-to-date, 1 skipped

Ready

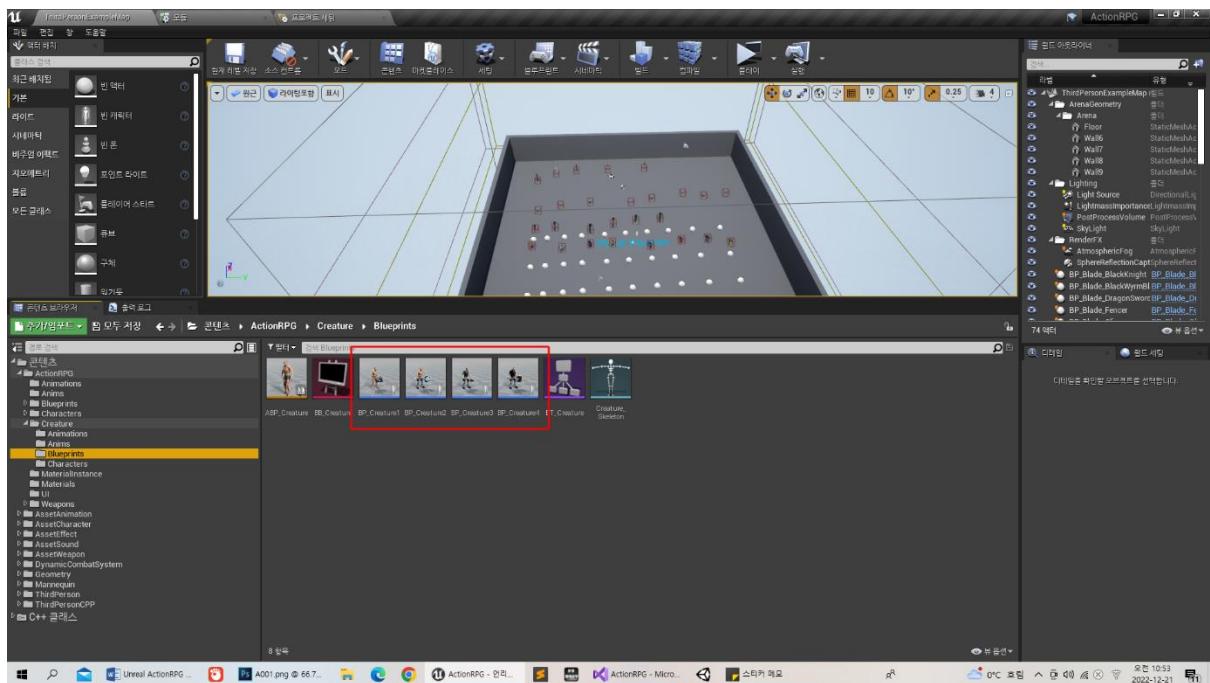
Unreal ActionRPG... A001.png @ 66.7... ActionRPG - 언리... ActionRPG - Micro... 스티커 메모

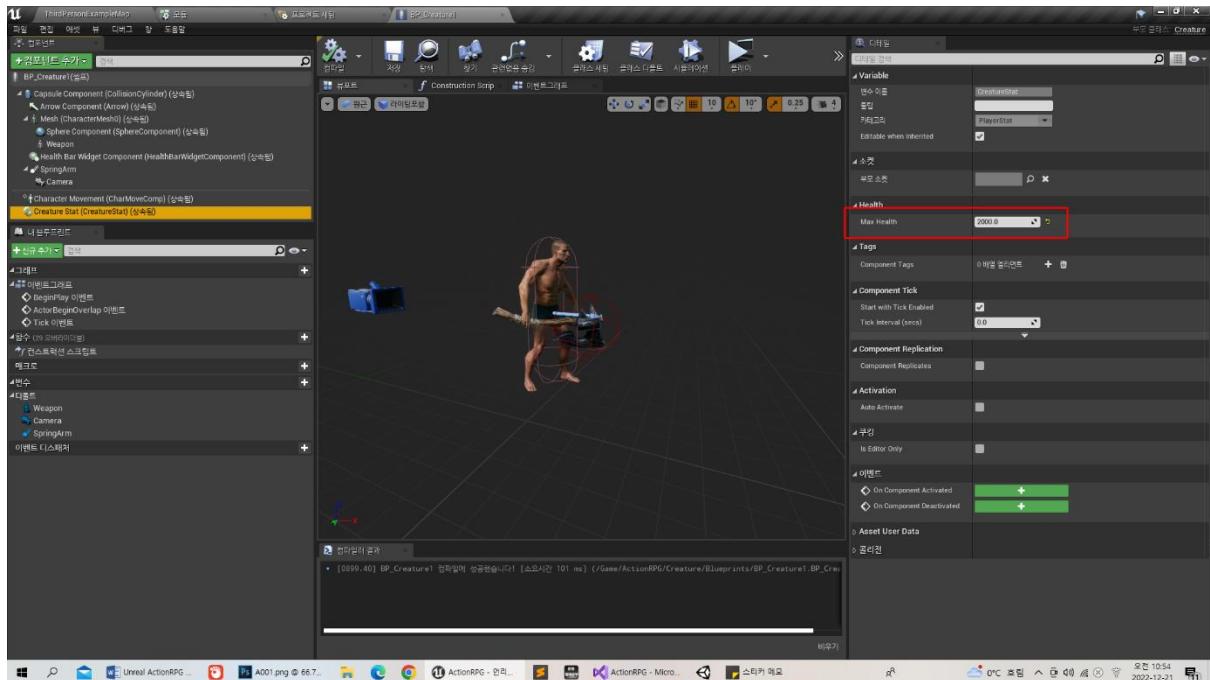
0xC 흐름 10:51 2022-12-21



```
102 // Called when the game starts or when spawned
103 void ACreature::BeginPlay()
104 {
105     Super::BeginPlay();
106
107     /** 크리처의 최대 체력값을 초기화 해 줍니다. */
108     //MaxHealth = 2000.0f;
109     /** 크리처의 체력값입니다. */
110     //Health = MaxHealth;
111     Health = CreatureStat->GetMaxHealth();
112
113     /** 체력바를 가져옵니다. */
114     auto const UserWidget = Cast<UHealthBar>(HealthBarWidgetComponent->GetUserWidgetObject());
115     /** 만일 체력바를 찾았다면? */
116     if (UserWidget)
117     {
118         /** 체력바에 현재 플레이어의 체력값을 표시합니다. */
119         UserWidget->SetHealthBarValue((Health / CreatureStat->GetMaxHealth() * 100) * 0.01f);
120     }
121 }
122
```

크리쳐 블루프린트에서 적용해 주도록 합니다.





플레이 해서 결과를 확인합니다.

