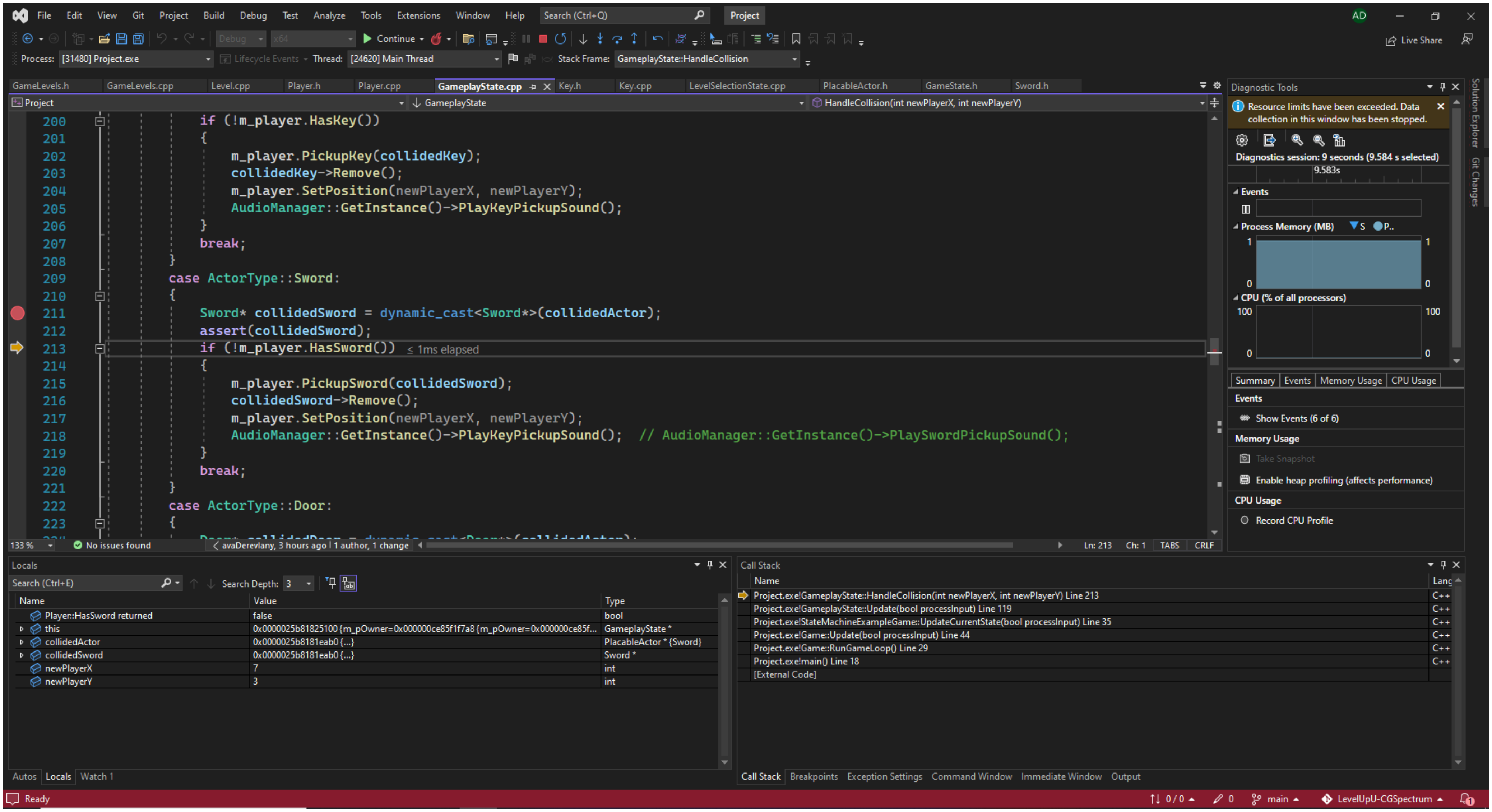


```
200                 if (!m_player.HasKey())
201                 {
202                     m_player.PickupKey(collidedKey);
203                     collidedKey->Remove();
204                     m_player.SetPosition(newPlayerX, newPlayerY);
205                     AudioManager::GetInstance()->PlayKeyPickupSound();
206                 }
207                 break;
208             }
209             case ActorType::Sword:
210             {
211                 Sword* collidedSword = dynamic_cast<Sword*>(collidedActor);
212                 assert(collidedSword);
213                 if (!m_player.HasSword())  ≤ 1ms elapsed
214                 {
215                     m_player.PickupSword(collidedSword);
216                     collidedSword->Remove();
217                     m_player.SetPosition(newPlayerX, newPlayerY);
218                     AudioManager::GetInstance()->PlayKeyPickupSound();  // AudioManager::GetInstance()->PlaySwordPickupSound();
219                 }
220                 break;
221             }
222             case ActorType::Door:
223             {
```

Resource limits have been exceeded. Data collection in this window has been stopped.

Diagnostics session: 9 seconds (9.584 s selected)

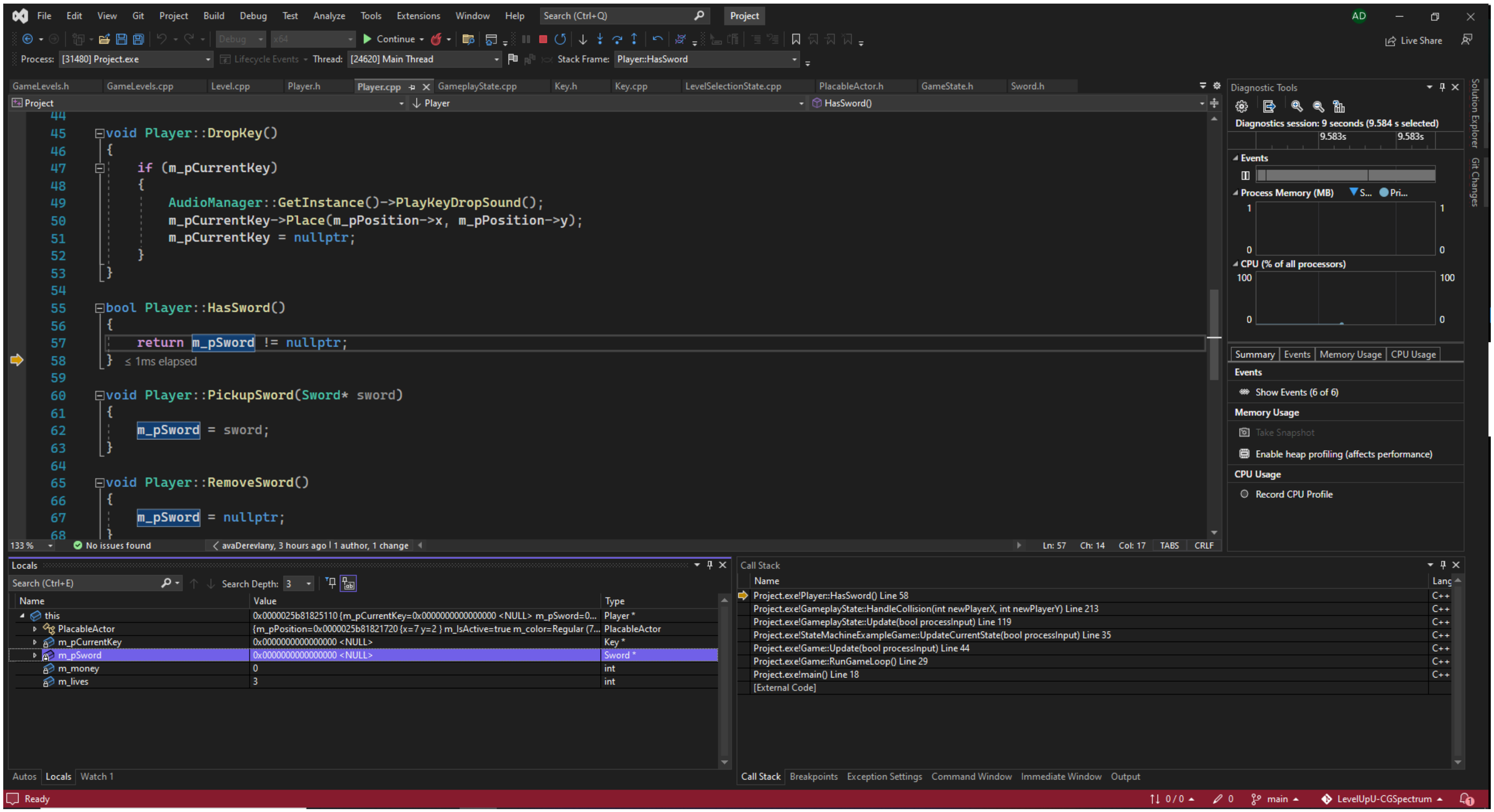
Locals

Name	Value	Type
Player::HasSword returned	false	bool
this	0x0000025b81825100 {m_pOwner=0x000000ce85f1f7a8 {m_pOwner=0x000000ce85f...	GameplayState *
collidedActor	0x0000025b8181eab0 {...}	PlacableActor * {Sword}
collidedSword	0x0000025b8181eab0 {...}	Sword *
newPlayerX	7	int
newPlayerY	3	int

Call Stack

Name	Lang
Project.exe!GameplayState::HandleCollision(int newPlayerX, int newPlayerY) Line 213	C++
Project.exe!GameplayState::Update(bool processInput) Line 119	C++
Project.exe!StateMachineExampleGame::UpdateCurrentState(bool processInput) Line 35	C++
Project.exe!Game::Update(bool processInput) Line 44	C++
Project.exe!Game::RunGameLoop() Line 29	C++
Project.exe!main() Line 18	C++
[External Code]	

When I was building my Sword actor, I wanted the sword to only last for the current level it was picked up on. Level 2 and Level 3 had swords on them. When you picked up the sword on Level 2, then progressed to Level 3 and tried to pick up the sword there, you originally could not. Line 213 “if (!m_player.HasSword()” would evaluate to true. The player still had the sword, even though I had thought that after each level, everything restarted. Had I set a breakpoint and compared the memory address of the collidedSword pointer to the sword pointer saved in the player (which I could check by stepping into the function HasSword(), shown in the image below) I would have seen that a) m_player.m_pSword was not null, and b) m_player.m_pSword had a different address then collidedSword. Although I did not do that originally, it could have helped me come to the conclusion quicker that I needed to remove the player’s sword pointer reference after each level, that the levels themselves reset, but the player is persistent.



```cpp
void Player::DropKey()
{
    if (m_pCurrentKey)
    {
        AudioManager::GetInstance()->PlayKeyDropSound();
        m_pCurrentKey->Place(m_pPosition->x, m_pPosition->y);
        m_pCurrentKey = nullptr;
    }
}

bool Player::HasSword()
{
    return m_pSword != nullptr;
}  ≤ 1ms elapsed

void Player::PickupSword(Sword* sword)
{
    m_pSword = sword;
}

void Player::RemoveSword()
{
    m_pSword = nullptr;
}
```

Locals

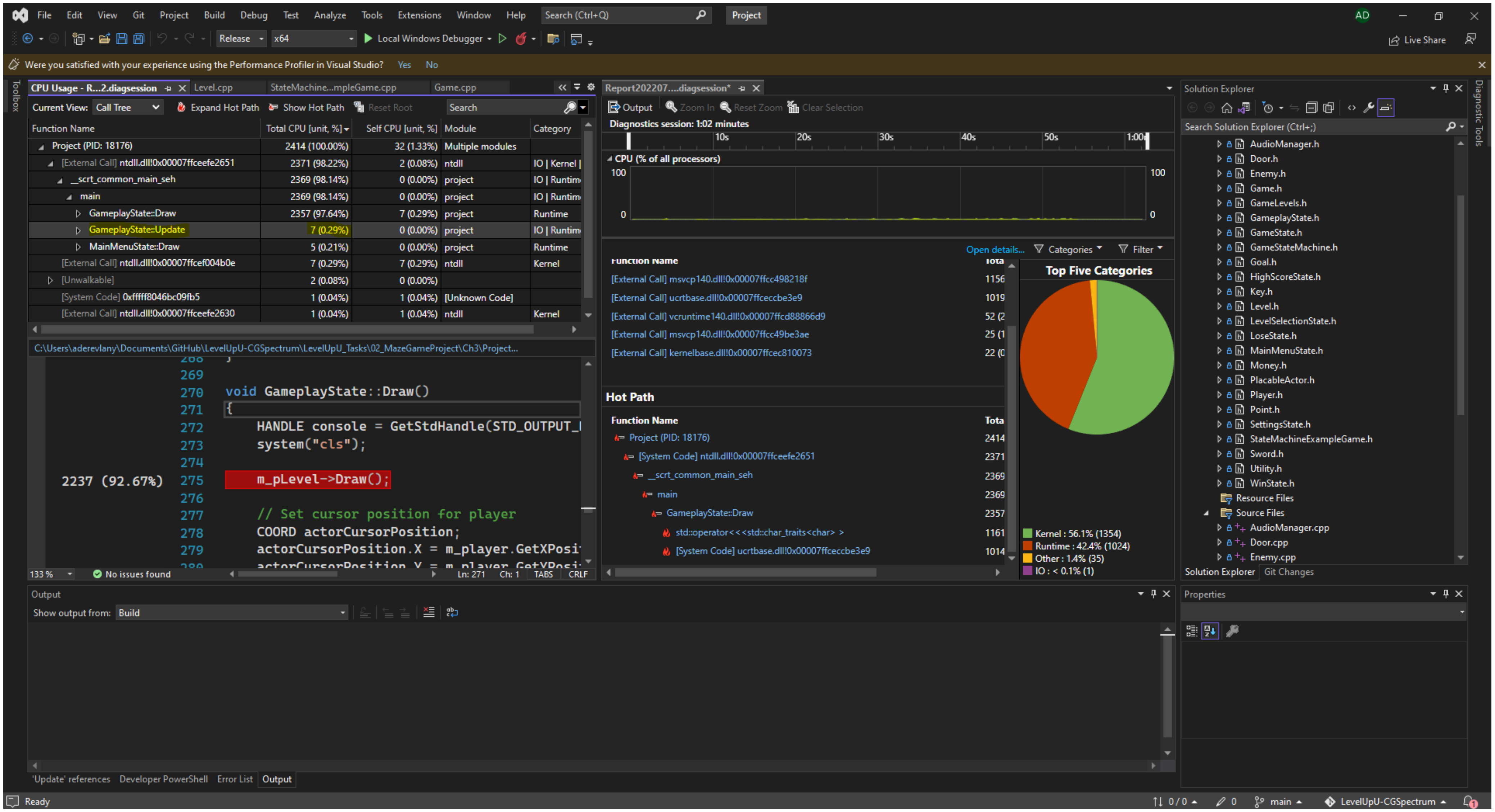
Name	Value	Type
this	0x0000025b81825110 {m_pCurrentKey=0x0000000000000000 <NULL> m_pSword=0...	Player *
PlacableActor	{m_pPosition=0x0000025b81821720 {x=7 y=2 } m_IsActive=true m_color=Regular (7...	PlacableActor
m_pCurrentKey	0x0000000000000000 <NULL>	Key *
m_pSword	0x0000000000000000 <NULL>	Sword *
m_money	0	int
m_lives	3	int

Call Stack

Name	Lang
Project.exe!Player::HasSword() Line 58	C++
Project.exe!GameplayState::HandleCollision(int newPlayerX, int newPlayerY) Line 213	C++
Project.exe!GameplayState::Update(bool processInput) Line 119	C++
Project.exe!StateMachineExampleGame::UpdateCurrentState(bool processInput) Line 35	C++
Project.exe!Game::Update(bool processInput) Line 44	C++
Project.exe!Game::RunGameLoop() Line 29	C++
Project.exe!main() Line 18	C++
[External Code]	

(the images above do not show the exact problem I was talking about as I had already solved it, but rather the player picking up a sword for the first time.)

Screenshot of the CPU performance, GameplayState::Update highlighted in yellow.



Were you satisfied with your experience using the Performance Profiler in Visual Studio? Yes No

Function Name	Total CPU [unit, %]	Self CPU [unit, %]	Module	Category
Project (PID: 18176)	2414 (100.00%)	32 (1.33%)	Multiple modules	
[External Call] ntdll.dll!0x00007ffceefe2651	2371 (98.22%)	2 (0.08%)	ntdll	IO | Kernel |
__scrt_common_main_seh	2369 (98.14%)	0 (0.00%)	project	IO | Runtim
main	2369 (98.14%)	0 (0.00%)	project	IO | Runtim
GameplayState::Draw	2357 (97.64%)	7 (0.29%)	project	Runtime
GameplayState::Update	7 (0.29%)	0 (0.00%)	project	IO | Runtim
MainMenuState::Draw	5 (0.21%)	0 (0.00%)	project	Runtime
[External Call] ntdll.dll!0x00007ffcef004b0e	7 (0.29%)	7 (0.29%)	ntdll	Kernel
[Unwalkable]	2 (0.08%)	0 (0.00%)		
[System Code] 0xfffff8046bc09fb5	1 (0.04%)	1 (0.04%)	[Unknown Code]	
[External Call] ntdll.dll!0x00007ffceefe2630	1 (0.04%)	1 (0.04%)	ntdll	Kernel

C:\Users\aderevlany\Documents\GitHub\LevelUpU-CGSpectrum\LevelUpU_Tasks\02_MazeGameProject\Ch3\Project...

```
            }

            void GameplayState::Draw()
            {
                HANDLE console = GetStdHandle(STD_OUTPUT_
                system("cls");

2237 (92.67%)   m_pLevel->Draw();

                // Set cursor position for player
                COORD actorCursorPosition;
                actorCursorPosition.X = m_player.GetXPosi
                actorCursorPosition.Y = m_player.GetYPosi
```

133 % No issues found Ln: 271 Ch: 1 TABS CRLF

Diagnostics session: 1:02 minutes

CPU (% of all processors)

Function Name	Tota
[External Call] msvcp140.dll!0x00007ffcc498218f	1156
[External Call] ucrtbase.dll!0x00007ffceccbe3e9	1019
[External Call] vcruntime140.dll!0x00007ffcd88866d9	52 (2
[External Call] msvcp140.dll!0x00007ffcc49be3ae	25 (1
[External Call] kernelbase.dll!0x00007ffcec810073	22 (0

Hot Path

Function Name	Tota
Project (PID: 18176)	2414
[System Code] ntdll.dll!0x00007ffceefe2651	2371
__scrt_common_main_seh	2369
main	2369
GameplayState::Draw	2357
std::operator<<<std::char_traits<char> >	1161
[System Code] ucrtbase.dll!0x00007ffceccbe3e9	1014

Top Five Categories

🟩 Kernel : 56.1% (1354)
🟥 Runtime : 42.4% (1024)
🟨 Other : 1.4% (35)
🟪 IO : < 0.1% (1)