Cheating My Way To Valhalla: Unity Hacking for Vikings

@Gr1mmie

player_entity

```
public class Player : Humanoid {
        string playerName = “Gr1mmie”;
        string playerDesc = “I like building tools and making things do things they shouldn’t”;
        string[] interests = { “Proud Sufferer of OOOHHHShinySyndrome”, “Malware Dev/Analysis + Rev
        Eng”, “Capabilities Dev”, “Gaming”,“Culinary Abominations”,  “Paintball” };
        string currentRole = “Pentester-For-Hire@RSM”;
        string prevRoles = { “R&D@FortyNorthSecurity”, “Intern@GuidePointSecurity” }
}
```

Agenda

1. Tools
2. Quick Intro to The Unity Engine
3. Quick Rundown of Common Cheats
4. Creating and Loading Unity Cheat
5. General Objectives When Writing Game Hacks
6. Hacking Valheim

Disclaimer (CMA)

I am not responsible for anything done with the information presented through this talk, blah blah blah. This is purely for educational purposes only and I do not promote nor do I condone the use of cheats in multiplayer games to gain an unfair advantage over other players. All cheats I write are used either only in single player mode, with bots, or with friends. I advise you do the same so you don't get banned etc.

Tools

- A general understanding of OOP languages and how they function/interact
- Enough knowledge of C# to read the code and hack together errrrr…"functionality"
- C# decompiler of your choosing, I like DNSpy
- IDE of your choosing, I am a madman and use Visual Studio
- Valheim (shocker, I know)
    - Or Unity game using Mono Framework, these techniques are effectively primitives and can be used in any mono game

Unity Engine

Intro To Unity Engine

- Game Engine allowing games to be created using C#
- Capable of creating 2D, 3D, VR/MR games
- Everything is an object, makes things way easier
- Serves as the engine for the following games
  - Batman: Arkham Shadow
  - Beat Saber
  - Cuphead
  - Escape from Tarkov
  - And many more

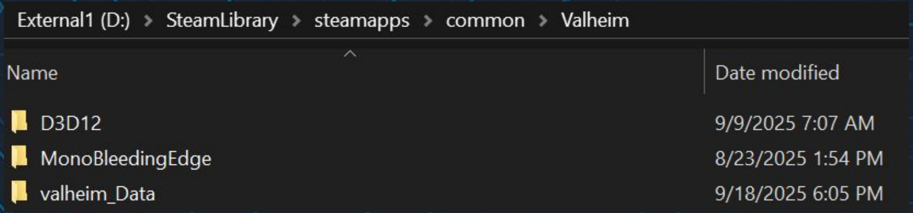
Unity Engine: Variants

Mono

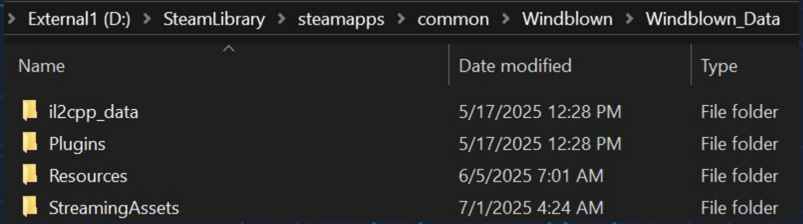
- Compiled just like any other C# assembly (JIT)
  - Decompiles…..just like any other C# assembly

IL2CPP

- Compiles to native binary (AOT)
  - Bit more of a pain to rev


External1 (D:) › SteamLibrary › steamapps › common › Valheim

Name	Date modified
D3D12	9/9/2025 7:07 AM
MonoBleedingEdge	8/23/2025 1:54 PM
valheim_Data	9/18/2025 6:05 PM


› External1 (D:) › SteamLibrary › steamapps › common › Windblown › Windblown_Data

Name	Date modified	Type
il2cpp_data	5/17/2025 12:28 PM	File folder
Plugins	5/17/2025 12:28 PM	File folder
Resources	6/5/2025 7:01 AM	File folder
StreamingAssets	7/1/2025 4:24 AM	File folder

Cheats

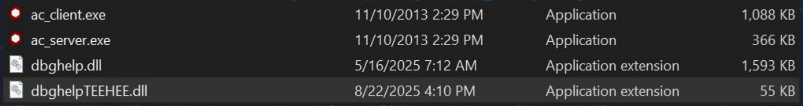
Cheat Types!

External

- Executed from outside of the game process, reaching within to access resources
- More limited than an internal cheat

Internal

- Loads within the game process, allowing much more access to resources
- Much easier to access internal pointers, functions, etc.



```
ac_client.exe        11/10/2013 2:29 PM   Application             1,088 KB
ac_server.exe        11/10/2013 2:29 PM   Application               366 KB
dbghelp.dll          5/16/2025 7:12 AM    Application extension   1,593 KB
dbghelpTEEHEE.dll    8/22/2025 4:10 PM    Application extension      55 KB
```

> Loading Module: dbghelp.dll

Property (Memory) Hacking

- Simplest form of a game cheat
- Allows cheat devs to modify a game value to whatever they see fit
- Want 99999 health/ammo/armor/etc. ? Done

Item Cheats

- Requires an understanding of how a game generates and utilizes items
- Implementation can vary quite a bit based on the game (or even engine used)

God Mode

- Usually involves writing in a custom function

- Few ways to approach this
  1. Patch the operation that deals damage
  2. “Freeze” health value at a set point
  3. Make it so that you never go below 1 HP
  4. ...or just run the SetGodMode() function?

ExtraSensory Perception (ESP)

Wall hax BABBYYYYYYYYYYYYY

- Requires understanding of the following
  - How the game handles entities + gaining access to the requisite entity list to gain access to the position of said entities
  - Calculating distance between current player and said entities
  - Working with the game's camera object
  - Casting 3D objects into a 2D visible space

- ESPs can be written to track all sorts of things including
  - enemies (wall hacks)
  - items
  - pretty much anything tracked by the game

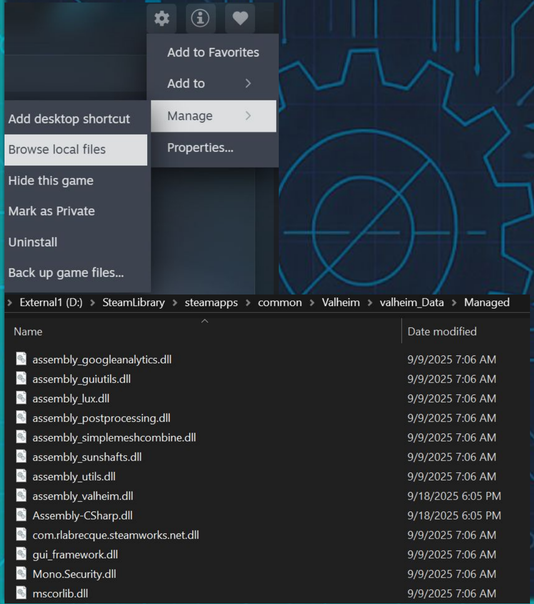
Teleportation Hacks

- You like skydiving? I don't....as a viking at least
- Involves modifying location coords of a player object
- ...Or just find and run the teleport function?

Unity Cheat Schematics

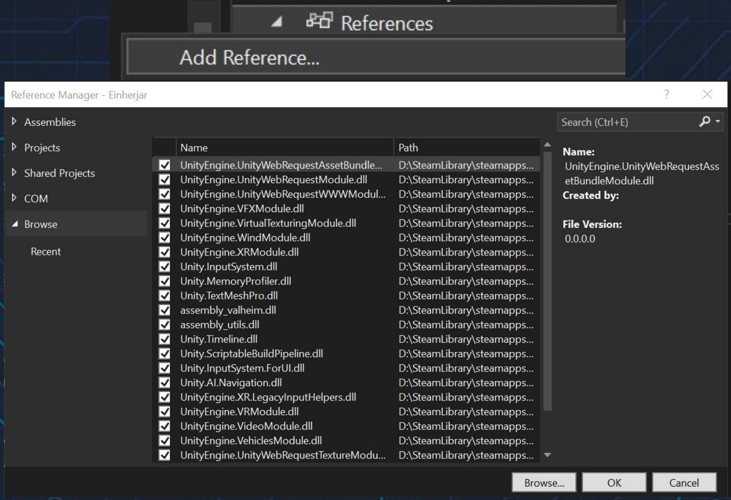
Importing Unity (& Game Functions)

1. Find game files (can be done via Browse local files on game page)
2. Copy-Paste all assembly*, Unity*, and UnityEngine* DLLs into Visual Studio References
    a. If you find an IMGUI DLL then grab that too, sometimes games use custom drawing libs



External1 (D:) › SteamLibrary › steamapps › common › Valheim › valheim_Data › Managed

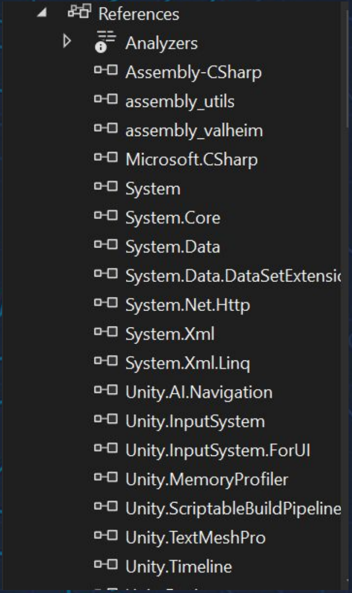
Name	Date modified
assembly_googleanalytics.dll	9/9/2025 7:06 AM
assembly_guiutils.dll	9/9/2025 7:06 AM
assembly_lux.dll	9/9/2025 7:06 AM
assembly_postprocessing.dll	9/9/2025 7:06 AM
assembly_simplemeshcombine.dll	9/9/2025 7:06 AM
assembly_sunshafts.dll	9/9/2025 7:06 AM
assembly_utils.dll	9/9/2025 7:06 AM
assembly_valheim.dll	9/18/2025 6:05 PM
Assembly-CSharp.dll	9/18/2025 6:05 PM
com.rlabrecque.steamworks.net.dll	9/9/2025 7:06 AM
gui_framework.dll	9/9/2025 7:06 AM
Mono.Security.dll	9/9/2025 7:06 AM
mscorlib.dll	9/9/2025 7:06 AM



	Name	Path
✓	UnityEngine.UnityWebRequestAssetBundle...	D:\SteamLibrary\steamapps...
✓	UnityEngine.UnityWebRequestModule.dll	D:\SteamLibrary\steamapps...
✓	UnityEngine.UnityWebRequestWWWModul...	D:\SteamLibrary\steamapps...
✓	UnityEngine.VFXModule.dll	D:\SteamLibrary\steamapps...
✓	UnityEngine.VirtualTexturingModule.dll	D:\SteamLibrary\steamapps...
✓	UnityEngine.WindModule.dll	D:\SteamLibrary\steamapps...
✓	UnityEngine.XRModule.dll	D:\SteamLibrary\steamapps...
✓	Unity.InputSystem.dll	D:\SteamLibrary\steamapps...
✓	Unity.MemoryProfiler.dll	D:\SteamLibrary\steamapps...
✓	Unity.TextMeshPro.dll	D:\SteamLibrary\steamapps...
✓	assembly_valheim.dll	D:\SteamLibrary\steamapps...
✓	assembly_utils.dll	D:\SteamLibrary\steamapps...
✓	Unity.Timeline.dll	D:\SteamLibrary\steamapps...
✓	Unity.ScriptableBuildPipeline.dll	D:\SteamLibrary\steamapps...
✓	Unity.InputSystem.ForUI.dll	D:\SteamLibrary\steamapps...
✓	Unity.AI.Navigation.dll	D:\SteamLibrary\steamapps...
✓	UnityEngine.XR.LegacyInputHelpers.dll	D:\SteamLibrary\steamapps...
✓	UnityEngine.VRModule.dll	D:\SteamLibrary\steamapps...
✓	UnityEngine.VideoModule.dll	D:\SteamLibrary\steamapps...
✓	UnityEngine.VehiclesModule.dll	D:\SteamLibrary\steamapps...
✓	UnityEngine.UnityWebRequestTextureModu...	D:\SteamLibrary\steamapps...

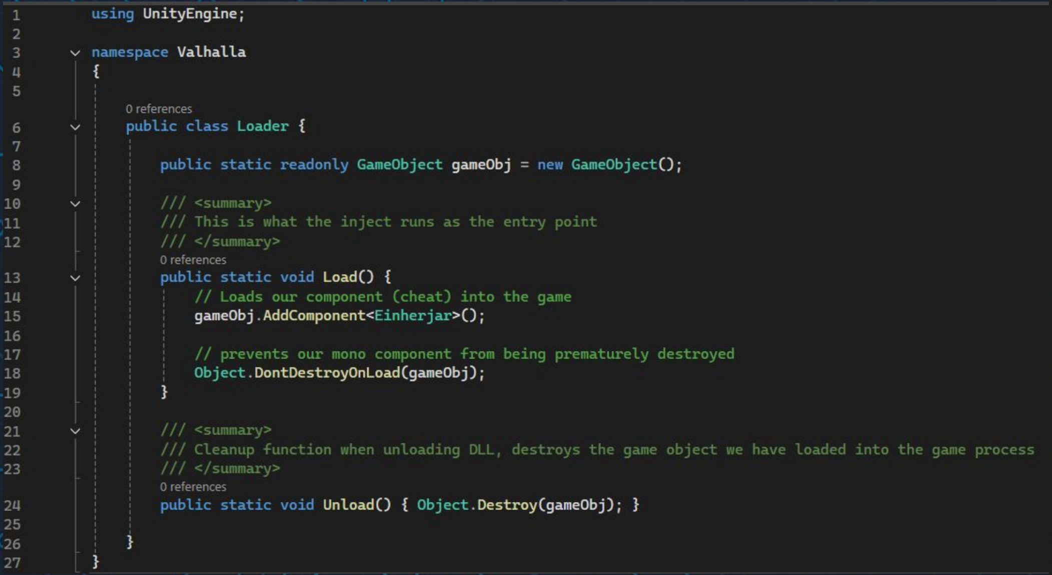
Name:
UnityEngine.UnityWebRequestAssetBundleModule.dll
Created by:

File Version:
0.0.0.0

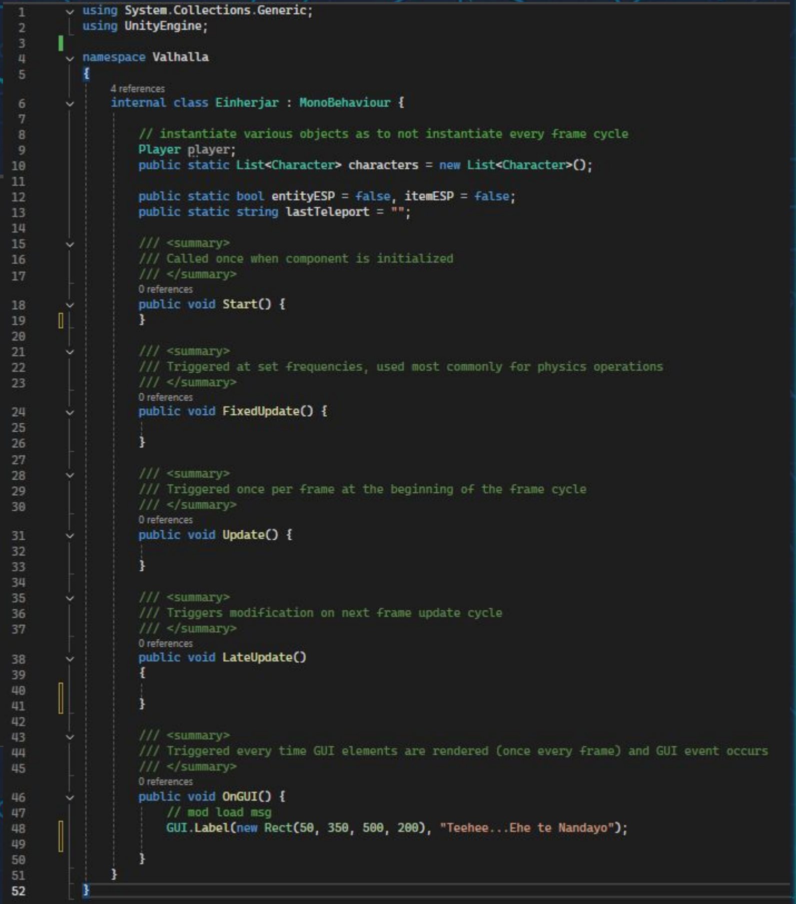


- References
  - Analyzers
  - Assembly-CSharp
  - assembly_utils
  - assembly_valheim
  - Microsoft.CSharp
  - System
  - System.Core
  - System.Data
  - System.Data.DataSetExtensi
  - System.Net.Http
  - System.Xml
  - System.Xml.Linq
  - Unity.AI.Navigation
  - Unity.InputSystem
  - Unity.InputSystem.ForUI
  - Unity.MemoryProfiler
  - Unity.ScriptableBuildPipeline
  - Unity.TextMeshPro
  - Unity.Timeline

Vikings First Cheat!



```csharp
using UnityEngine;

namespace Valhalla
{

    0 references
    public class Loader {

        public static readonly GameObject gameObj = new GameObject();

        /// <summary>
        /// This is what the inject runs as the entry point
        /// </summary>
        0 references
        public static void Load() {
            // Loads our component (cheat) into the game
            gameObj.AddComponent<Einherjar>();

            // prevents our mono component from being prematurely destroyed
            Object.DontDestroyOnLoad(gameObj);
        }

        /// <summary>
        /// Cleanup function when unloading DLL, destroys the game object we have loaded into the game process
        /// </summary>
        0 references
        public static void Unload() { Object.Destroy(gameObj); }

    }
}
```



```csharp
using System.Collections.Generic;
using UnityEngine;

namespace Valhalla
{
    4 references
    internal class Einherjar : MonoBehaviour {

        // instantiate various objects as to not instantiate every frame cycle
        Player player;
        public static List<Character> characters = new List<Character>();

        public static bool entityESP = false, itemESP = false;
        public static string lastTeleport = "";

        /// <summary>
        /// Called once when component is initialized
        /// </summary>
        0 references
        public void Start() {
        }

        /// <summary>
        /// Triggered at set frequencies, used most commonly for physics operations
        /// </summary>
        0 references
        public void FixedUpdate() {

        }

        /// <summary>
        /// Triggered once per frame at the beginning of the frame cycle
        /// </summary>
        0 references
        public void Update() {

        }

        /// <summary>
        /// Triggers modification on next frame update cycle
        /// </summary>
        0 references
        public void LateUpdate()
        {

        }

        /// <summary>
        /// Triggered every time GUI elements are rendered (once every frame) and GUI event occurs
        /// </summary>
        0 references
        public void OnGUI() {
            // mod load msg
            GUI.Label(new Rect(50, 350, 500, 200), "Teehee...Ehe te Nandayo");

        }
    }
}
```

Unity Engine Game Flow

- Executes a sequence of functions determining actions to be performed each "frame cycle" (™ not really)

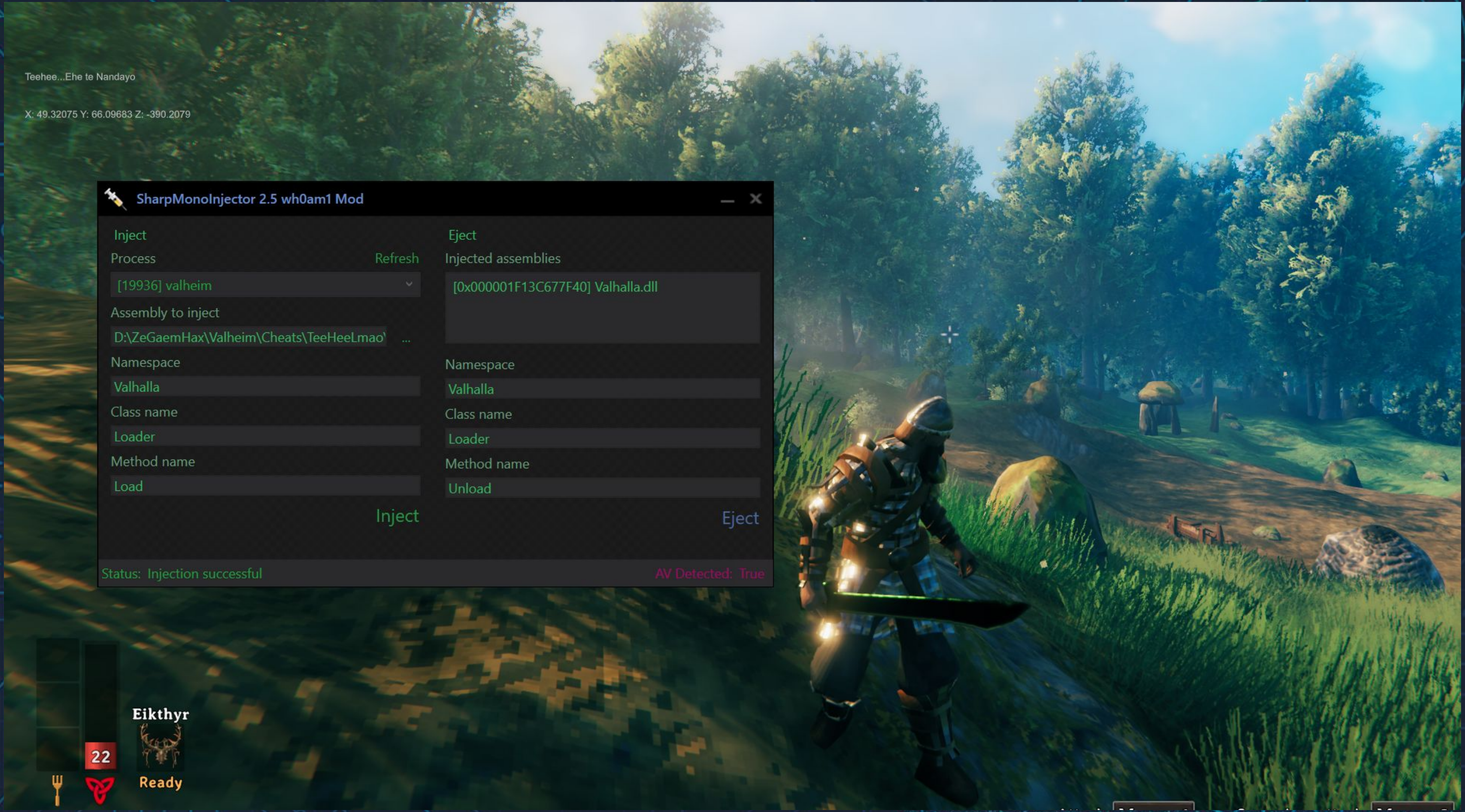
- "frame cycle"?
  - A full loop of all operations performed to generate/operate on a frame (gameplay, physics, draw calls, etc.)
  - FPS = how many frame cycles per second. This is why higher frame rate is correlated with better performance

Unity Engine Game Flow: Key Functions

- Awake()
  - Runs at start of object instantiation before all other functions
- Start()
  - Triggers once before any update functions are called
- FixedUpdate()
  - Part of physics loop, is called at fixed intervals. Can be called multiple times per frame
- Update()
  - Triggers on every frame cycle
  - Code is executed at the start of frame cycle
- LateUpdate()
  - Triggered after all other update calls have finished in a frame cycle
  - Code is executed in next frame cycle
- OnGUI()
  - Triggers everytime a GUI element is rendered and a GUI event occurs
  - Only function capable of rendering/performing GUI actions…sweet mother of lag

https://docs.unity3d.com/ScriptReference/MonoBehaviour.html

Loading Our Unity Cheat



Teehee...Ehe te Nandayo

X: 49.32075 Y: 66.09683 Z: -390.2079

SharpMonoInjector 2.5 wh0am1 Mod

Inject		Eject
Process	Refresh	Injected assemblies
[19936] valheim		[0x000001F13C677F40] Valhalla.dll
Assembly to inject		
D:\ZeGaemHax\Valheim\Cheats\TeeHeeLmao'	...	
Namespace		Namespace
Valhalla		Valhalla
Class name		Class name
Loader		Loader
Method name		Method name
Load		Unload
Inject		Eject

Status: Injection successful

AV Detected: True

https://github.com/warbler/SharpMonoInjector

Game Hacking Objectives

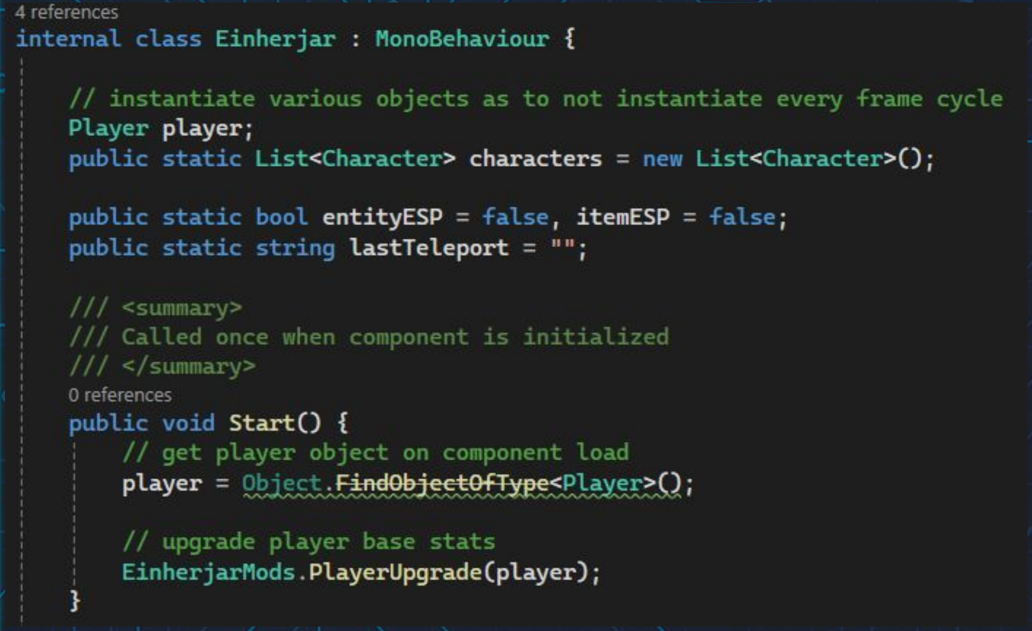
- Obtaining player object – pretty much Domain Admin/Global Admin/etc.
  - Can be used to access weapon/item, player stats, location, etc.

- Reversing enough of game to replicate/create functionality
  - I.e finding Ammo decrement code and patching so that ammo is not depleted when shooting

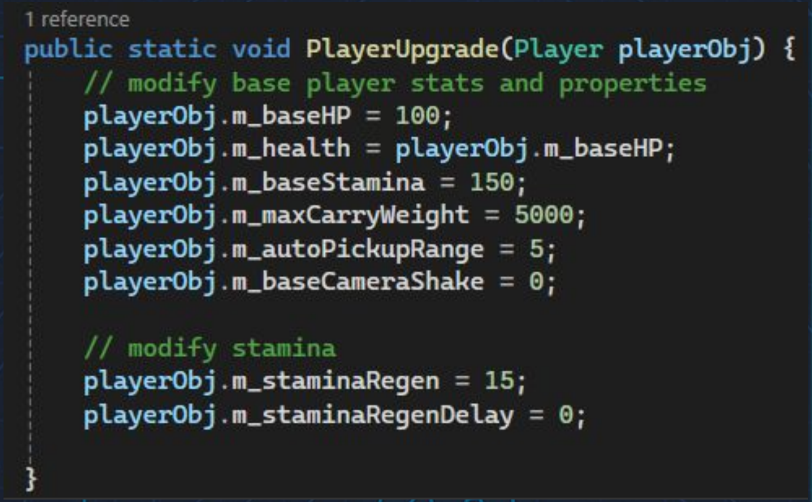
- Accessing requisite game objects/internal functions

Obtaining Player Object

- The DA of Game Hacking…kinda
  - Provides access to all properties tied to player object
  - Serves as the base for building player-related cheats (think ammo/weapon mods, invincibility, etc.)



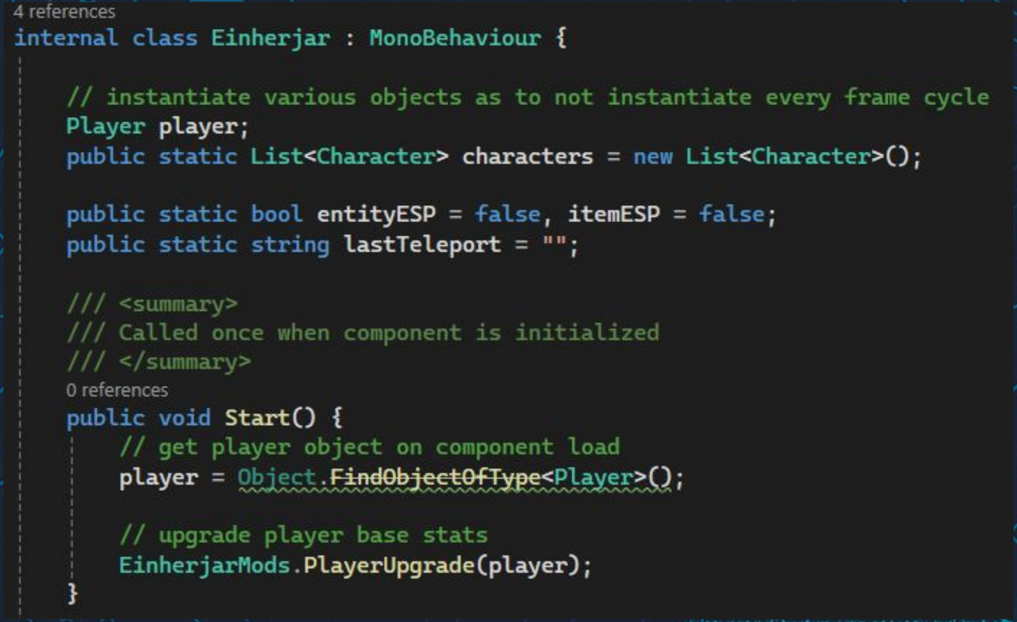
```
4 references
internal class Einherjar : MonoBehaviour {

    // instantiate various objects as to not instantiate every frame cycle
    Player player;
    public static List<Character> characters = new List<Character>();

    public static bool entityESP = false, itemESP = false;
    public static string lastTeleport = "";

    /// <summary>
    /// Called once when component is initialized
    /// </summary>
    0 references
    public void Start() {
        // get player object on component load
        player = Object.FindObjectOfType<Player>();

        // upgrade player base stats
        EinherjarMods.PlayerUpgrade(player);
    }
```



```
1 reference
public static void PlayerUpgrade(Player playerObj) {
    // modify base player stats and properties
    playerObj.m_baseHP = 100;
    playerObj.m_health = playerObj.m_baseHP;
    playerObj.m_baseStamina = 150;
    playerObj.m_maxCarryWeight = 5000;
    playerObj.m_autoPickupRange = 5;
    playerObj.m_baseCameraShake = 0;

    // modify stamina
    playerObj.m_staminaRegen = 15;
    playerObj.m_staminaRegenDelay = 0;

}
```

But first…Mono Objects

- Throw back to OOP!
- This is C#...so everything is an object
    - If everything is an object, then we have to fetch it as an object



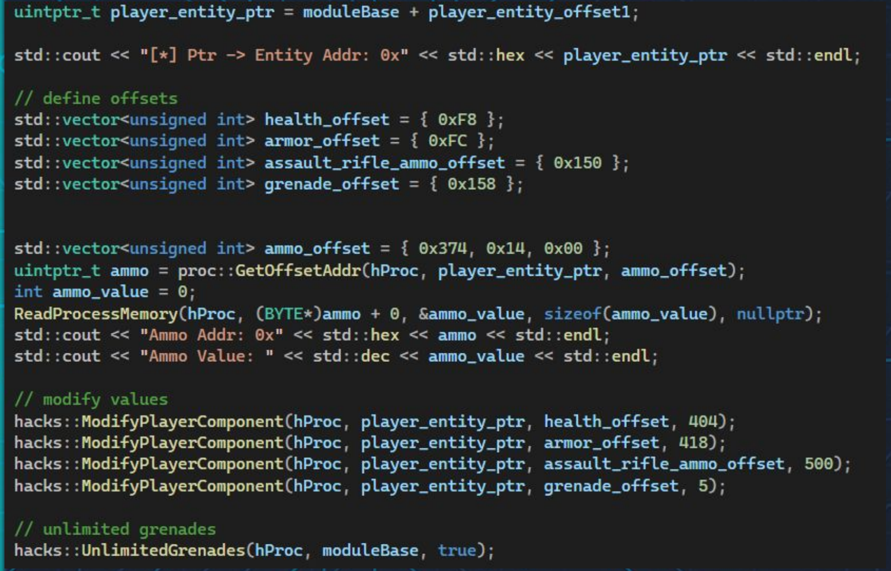
```
4 references
internal class Einherjar : MonoBehaviour {

    // instantiate various objects as to not instantiate every frame cycle
    Player player;
    public static List<Character> characters = new List<Character>();

    public static bool entityESP = false, itemESP = false;
    public static string lastTeleport = "";

    /// <summary>
    /// Called once when component is initialized
    /// </summary>
    0 references
    public void Start() {
        // get player object on component load
        player = Object.FindObjectOfType<Player>();

        // upgrade player base stats
        EinherjarMods.PlayerUpgrade(player);
    }
```

https://docs.unity3d.com/6000.2/Documentation/ScriptReference/Object.FindObjectsByType.html

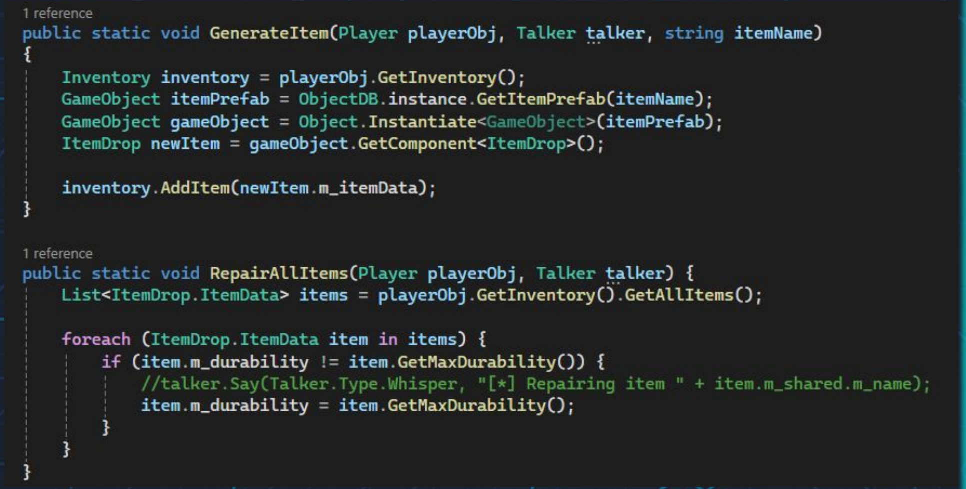
Offset this, Offset that, Offset …ah nevermind

- Approach to hacking native games: find offset -> validate offset -> use for cheats

- Unity games == OOP…so just fetch object/property



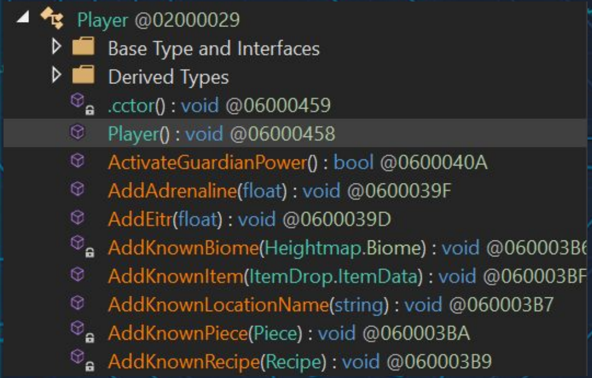
```cpp
uintptr_t player_entity_ptr = moduleBase + player_entity_offset1;

std::cout << "[*] Ptr -> Entity Addr: 0x" << std::hex << player_entity_ptr << std::endl;

// define offsets
std::vector<unsigned int> health_offset = { 0xF8 };
std::vector<unsigned int> armor_offset = { 0xFC };
std::vector<unsigned int> assault_rifle_ammo_offset = { 0x150 };
std::vector<unsigned int> grenade_offset = { 0x158 };


std::vector<unsigned int> ammo_offset = { 0x374, 0x14, 0x00 };
uintptr_t ammo = proc::GetOffsetAddr(hProc, player_entity_ptr, ammo_offset);
int ammo_value = 0;
ReadProcessMemory(hProc, (BYTE*)ammo + 0, &ammo_value, sizeof(ammo_value), nullptr);
std::cout << "Ammo Addr: 0x" << std::hex << ammo << std::endl;
std::cout << "Ammo Value: " << std::dec << ammo_value << std::endl;

// modify values
hacks::ModifyPlayerComponent(hProc, player_entity_ptr, health_offset, 404);
hacks::ModifyPlayerComponent(hProc, player_entity_ptr, armor_offset, 418);
hacks::ModifyPlayerComponent(hProc, player_entity_ptr, assault_rifle_ammo_offset, 500);
hacks::ModifyPlayerComponent(hProc, player_entity_ptr, grenade_offset, 5);

// unlimited grenades
hacks::UnlimitedGrenades(hProc, moduleBase, true);
```



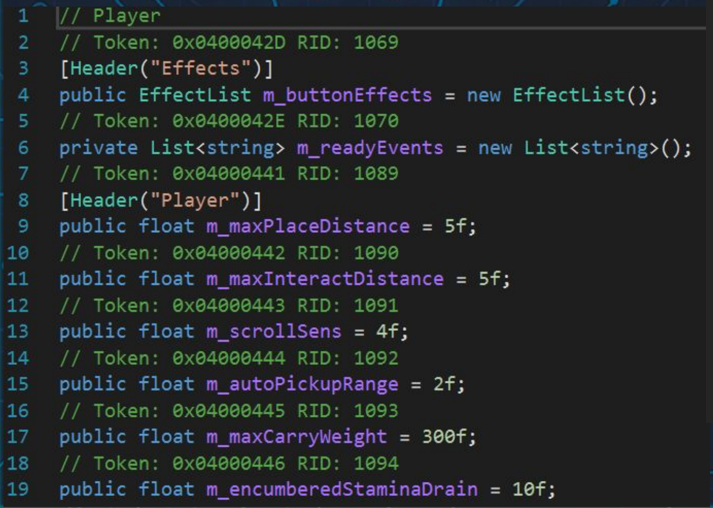
```csharp
1 reference
public static void GenerateItem(Player playerObj, Talker talker, string itemName)
{
    Inventory inventory = playerObj.GetInventory();
    GameObject itemPrefab = ObjectDB.instance.GetItemPrefab(itemName);
    GameObject gameObject = Object.Instantiate<GameObject>(itemPrefab);
    ItemDrop newItem = gameObject.GetComponent<ItemDrop>();

    inventory.AddItem(newItem.m_itemData);
}

1 reference
public static void RepairAllItems(Player playerObj, Talker talker) {
    List<ItemDrop.ItemData> items = playerObj.GetInventory().GetAllItems();

    foreach (ItemDrop.ItemData item in items) {
        if (item.m_durability != item.GetMaxDurability()) {
            //talker.Say(Talker.Type.Whisper, "[*] Repairing item " + item.m_shared.m_name);
            item.m_durability = item.GetMaxDurability();
        }
    }
}
```

Einherjar: To Valhalla We Ascend

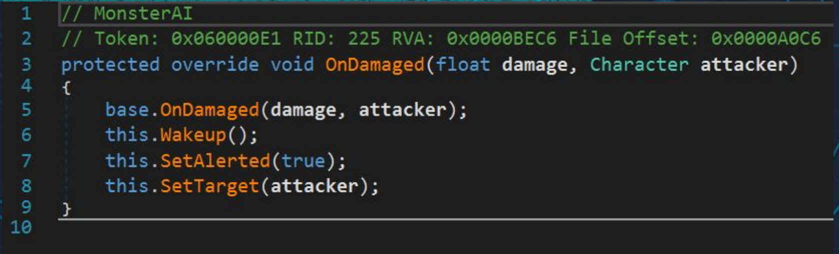
Reversing Valheim



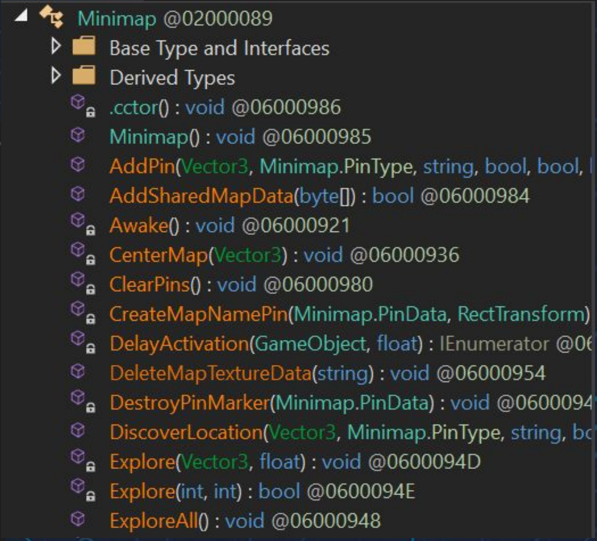
```
◢ Player @02000029
  ▷ Base Type and Interfaces
  ▷ Derived Types
    .cctor() : void @06000459
    Player() : void @06000458
    ActivateGuardianPower() : bool @0600040A
    AddAdrenaline(float) : void @0600039F
    AddEitr(float) : void @0600039D
    AddKnownBiome(Heightmap.Biome) : void @060003B6
    AddKnownItem(ItemDrop.ItemData) : void @060003BF
    AddKnownLocationName(string) : void @060003B7
    AddKnownPiece(Piece) : void @060003BA
    AddKnownRecipe(Recipe) : void @060003B9
```



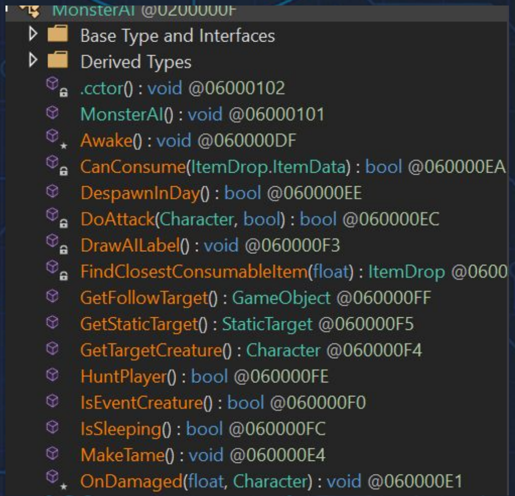
```
// Player
// Token: 0x0400042D RID: 1069
[Header("Effects")]
public EffectList m_buttonEffects = new EffectList();
// Token: 0x0400042E RID: 1070
private List<string> m_readyEvents = new List<string>();
// Token: 0x04000441 RID: 1089
[Header("Player")]
public float m_maxPlaceDistance = 5f;
// Token: 0x04000442 RID: 1090
public float m_maxInteractDistance = 5f;
// Token: 0x04000443 RID: 1091
public float m_scrollSens = 4f;
// Token: 0x04000444 RID: 1092
public float m_autoPickupRange = 2f;
// Token: 0x04000445 RID: 1093
public float m_maxCarryWeight = 300f;
// Token: 0x04000446 RID: 1094
public float m_encumberedStaminaDrain = 10f;
```



```
// MonsterAI
// Token: 0x060000E1 RID: 225 RVA: 0x0000BEC6 File Offset: 0x0000A0C6
protected override void OnDamaged(float damage, Character attacker)
{
    base.OnDamaged(damage, attacker);
    this.Wakeup();
    this.SetAlerted(true);
    this.SetTarget(attacker);
}
```

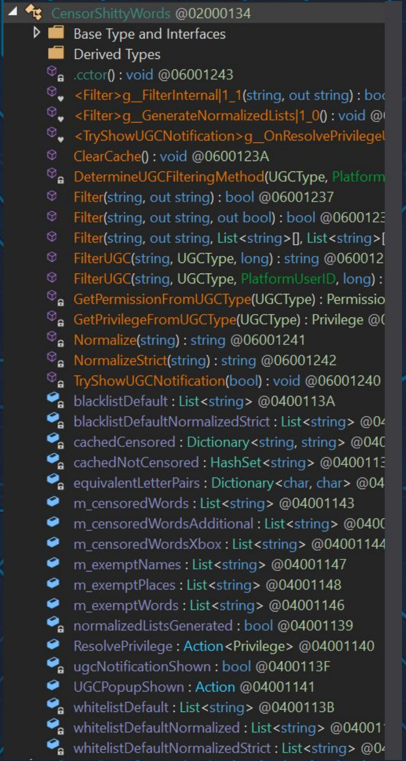


```
◢ Minimap @02000089
  ▷ Base Type and Interfaces
  ▷ Derived Types
    .cctor() : void @06000986
    Minimap() : void @06000985
    AddPin(Vector3, Minimap.PinType, string, bool, bool, l
    AddSharedMapData(byte[]) : bool @06000984
    Awake() : void @06000921
    CenterMap(Vector3) : void @06000936
    ClearPins() : void @06000980
    CreateMapNamePin(Minimap.PinData, RectTransform)
    DelayActivation(GameObject, float) : IEnumerator @06
    DeleteMapTextureData(string) : void @06000954
    DestroyPinMarker(Minimap.PinData) : void @0600094
    DiscoverLocation(Vector3, Minimap.PinType, string, bo
    Explore(Vector3, float) : void @0600094D
    Explore(int, int) : bool @0600094E
    ExploreAll() : void @06000948
```

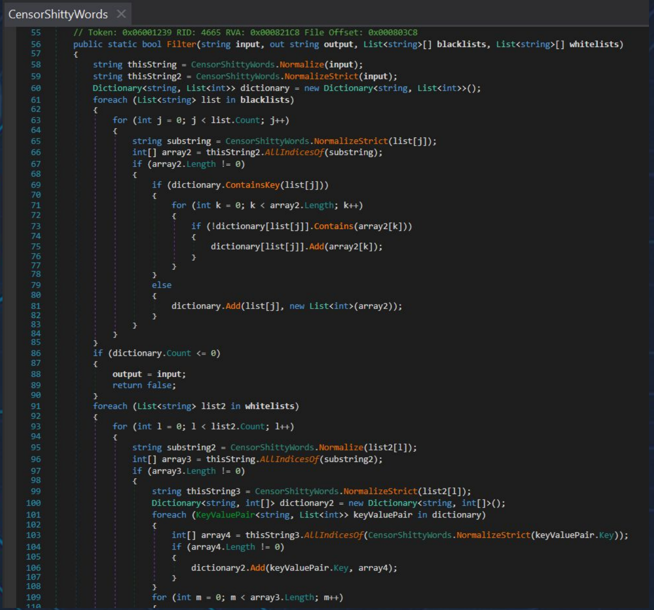


```
  MonsterAI @0200000F
  ▷ Base Type and Interfaces
  ▷ Derived Types
    .cctor() : void @06000102
    MonsterAI() : void @06000101
    Awake() : void @060000DF
    CanConsume(ItemDrop.ItemData) : bool @060000EA
    DespawnInDay() : bool @060000EE
    DoAttack(Character, bool) : bool @060000EC
    DrawAILabel() : void @060000F3
    FindClosestConsumableItem(float) : ItemDrop @0600
    GetFollowTarget() : GameObject @060000FF
    GetStaticTarget() : StaticTarget @060000F5
    GetTargetCreature() : Character @060000F4
    HuntPlayer() : bool @060000FE
    IsEventCreature() : bool @060000F0
    IsSleeping() : bool @060000FC
    MakeTame() : void @060000E4
    OnDamaged(float, Character) : void @060000E1
```

Reversing Valheim: Funny Things I’ve Found

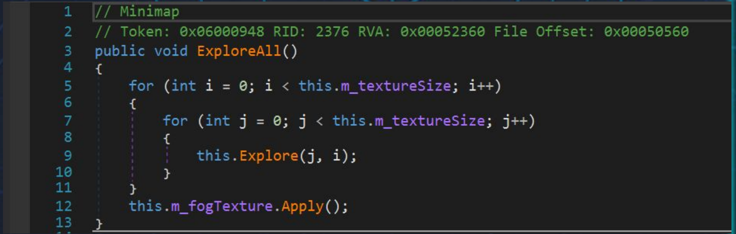


```
CensorShittyWords @02000134
  Base Type and Interfaces
  Derived Types
  .cctor() : void @06001243
  <Filter>g__FilterInternal|1_1(string, out string) : boo
  <Filter>g__GenerateNormalizedLists|1_0() : void @
  <TryShowUGCNotification>g__OnResolvePrivilege
  ClearCache() : void @0600123A
  DetermineUGCFilteringMethod(UGCType, Platform
  Filter(string, out string) : bool @06001237
  Filter(string, out string, out bool) : bool @0600123
  Filter(string, out string, List<string>[], List<string>[
  FilterUGC(string, UGCType, long) : string @060012
  FilterUGC(string, UGCType, PlatformUserID, long) :
  GetPermissionFromUGCType(UGCType) : Permissio
  GetPrivilegeFromUGCType(UGCType) : Privilege @
  Normalize(string) : string @06001241
  NormalizeStrict(string) : string @06001242
  TryShowUGCNotification(bool) : void @06001240
  blacklistDefault : List<string> @0400113A
  blacklistDefaultNormalizedStrict : List<string> @04
  cachedCensored : Dictionary<string, string> @040
  cachedNotCensored : HashSet<string> @040011
  equivalentLetterPairs : Dictionary<char, char> @04
  m_censoredWords : List<string> @04001143
  m_censoredWordsAdditional : List<string> @040
  m_censoredWordsXbox : List<string> @04001144
  m_exemptNames : List<string> @04001147
  m_exemptPlaces : List<string> @04001148
  m_exemptWords : List<string> @04001146
  normalizedListsGenerated : bool @04001139
  ResolvePrivilege : Action<Privilege> @04001140
  ugcNotificationShown : bool @0400113F
  UGCPopupShown : Action @04001141
  whitelistDefault : List<string> @0400113B
  whitelistDefaultNormalized : List<string> @04001
  whitelistDefaultNormalizedStrict : List<string> @04
```

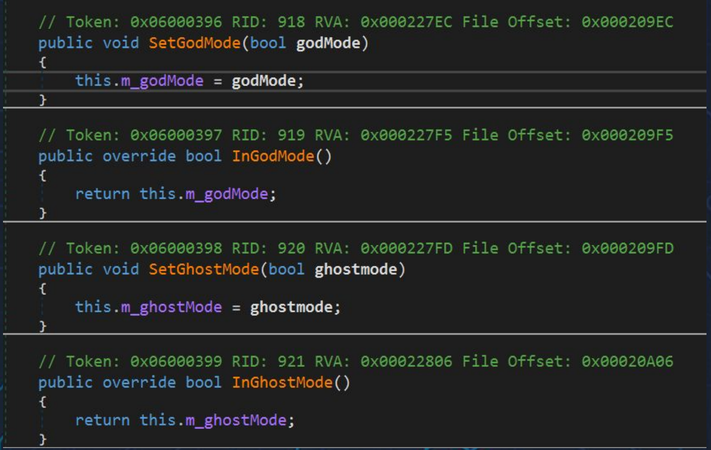


CensorShittyWords

```csharp
// Token: 0x06001239 RID: 4665 RVA: 0x000821C8 File Offset: 0x000803C8
public static bool Filter(string input, out string output, List<string>[] blacklists, List<string>[] whitelists)
{
    string thisString = CensorShittyWords.Normalize(input);
    string thisString2 = CensorShittyWords.NormalizeStrict(input);
    Dictionary<string, List<int>> dictionary = new Dictionary<string, List<int>>();
    foreach (List<string> list in blacklists)
    {
        for (int j = 0; j < list.Count; j++)
        {
            string substring = CensorShittyWords.NormalizeStrict(list[j]);
            int[] array2 = thisString2.AllIndicesOf(substring);
            if (array2.Length != 0)
            {
                if (dictionary.ContainsKey(list[j]))
                {
                    for (int k = 0; k < array2.Length; k++)
                    {
                        if (!dictionary[list[j]].Contains(array2[k]))
                        {
                            dictionary[list[j]].Add(array2[k]);
                        }
                    }
                }
                else
                {
                    dictionary.Add(list[j], new List<int>(array2));
                }
            }
        }
    }
    if (dictionary.Count <= 0)
    {
        output = input;
        return false;
    }
    foreach (List<string> list2 in whitelists)
    {
        for (int l = 0; l < list2.Count; l++)
        {
            string substring2 = CensorShittyWords.Normalize(list2[l]);
            int[] array3 = thisString.AllIndicesOf(substring2);
            if (array3.Length != 0)
            {
                string thisString3 = CensorShittyWords.NormalizeStrict(list2[l]);
                Dictionary<string, int[]> dictionary2 = new Dictionary<string, int[]>();
                foreach (KeyValuePair<string, List<int>> keyValuePair in dictionary)
                {
                    int[] array4 = thisString3.AllIndicesOf(CensorShittyWords.NormalizeStrict(keyValuePair.Key));
                    if (array4.Length != 0)
                    {
                        dictionary2.Add(keyValuePair.Key, array4);
                    }
                }
                for (int m = 0; m < array3.Length; m++)
```

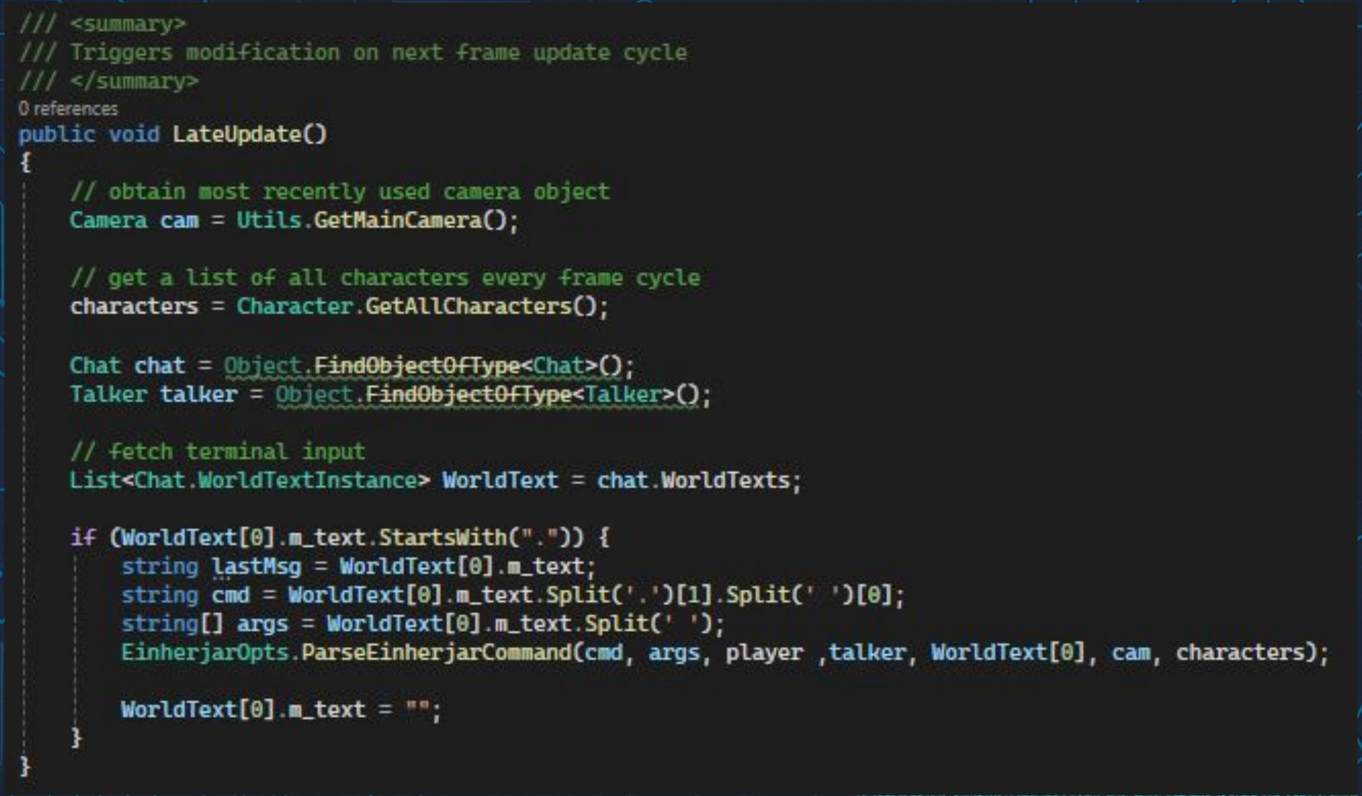


```csharp
// Minimap
// Token: 0x06000948 RID: 2376 RVA: 0x00052360 File Offset: 0x00050560
public void ExploreAll()
{
    for (int i = 0; i < this.m_textureSize; i++)
    {
        for (int j = 0; j < this.m_textureSize; j++)
        {
            this.Explore(j, i);
        }
    }
    this.m_fogTexture.Apply();
}
```



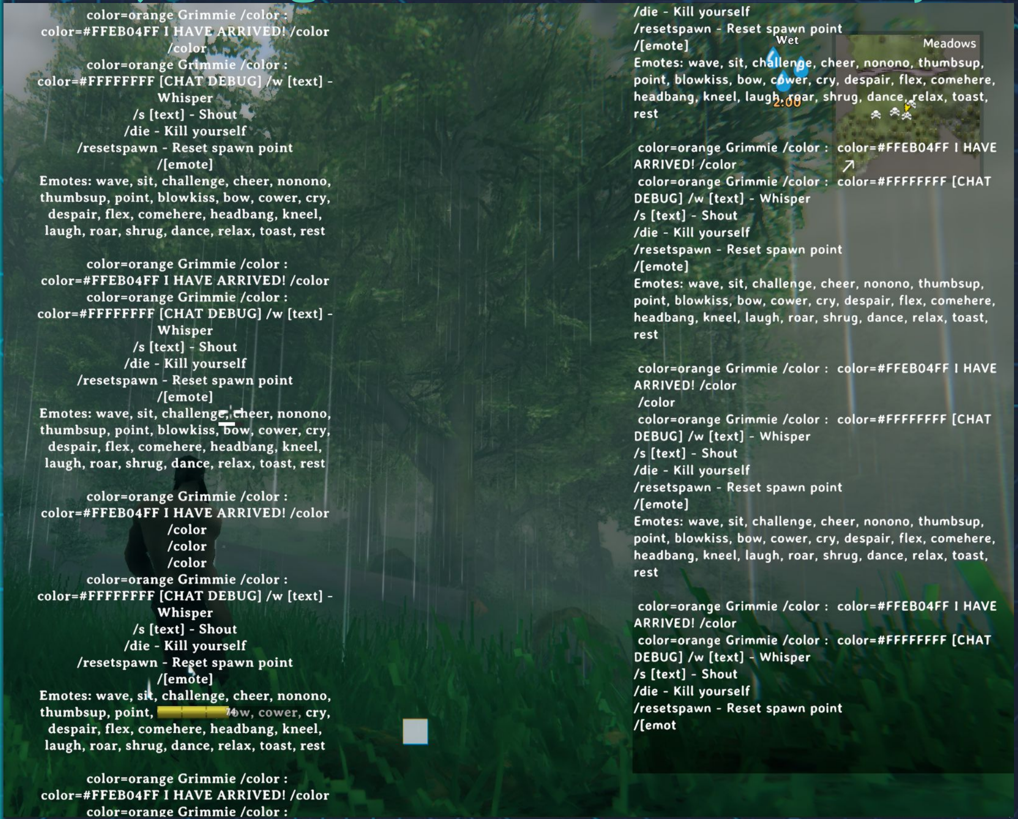
```csharp
// Token: 0x06000396 RID: 918 RVA: 0x000227EC File Offset: 0x000209EC
public void SetGodMode(bool godMode)
{
    this.m_godMode = godMode;
}

// Token: 0x06000397 RID: 919 RVA: 0x000227F5 File Offset: 0x000209F5
public override bool InGodMode()
{
    return this.m_godMode;
}

// Token: 0x06000398 RID: 920 RVA: 0x000227FD File Offset: 0x000209FD
public void SetGhostMode(bool ghostmode)
{
    this.m_ghostMode = ghostmode;
}

// Token: 0x06000399 RID: 921 RVA: 0x00022806 File Offset: 0x00020A06
public override bool InGhostMode()
{
    return this.m_ghostMode;
}
```

Hijacking world_text



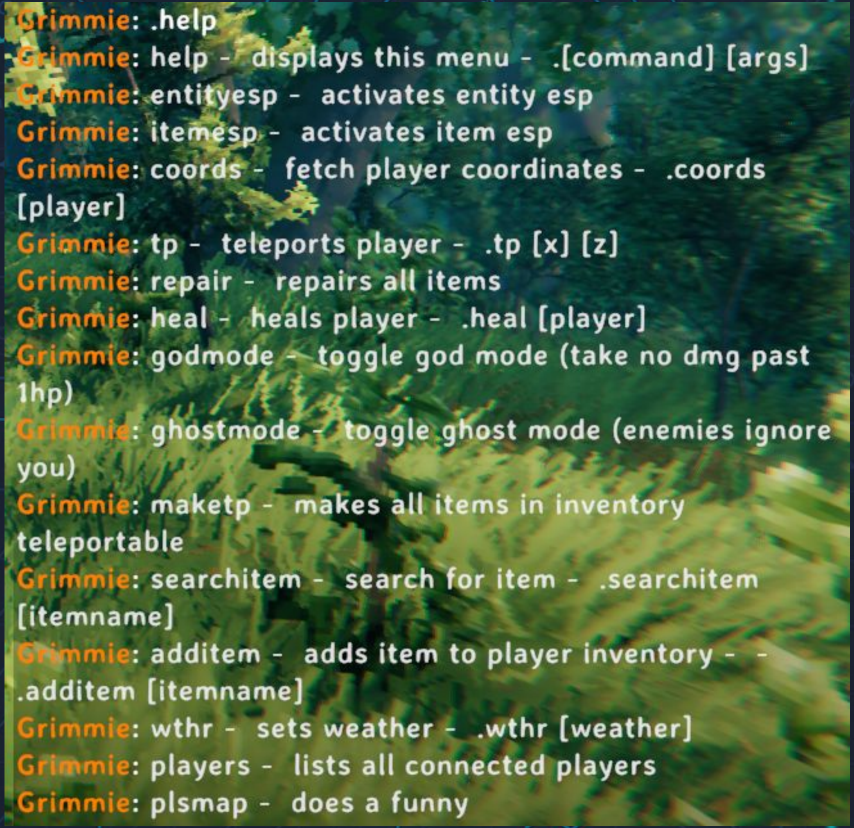
```
/// <summary>
/// Triggers modification on next frame update cycle
/// </summary>
0 references
public void LateUpdate()
{
    // obtain most recently used camera object
    Camera cam = Utils.GetMainCamera();

    // get a list of all characters every frame cycle
    characters = Character.GetAllCharacters();

    Chat chat = Object.FindObjectOfType<Chat>();
    Talker talker = Object.FindObjectOfType<Talker>();

    // fetch terminal input
    List<Chat.WorldTextInstance> WorldText = chat.WorldTexts;

    if (WorldText[0].m_text.StartsWith(".")) {
        string lastMsg = WorldText[0].m_text;
        string cmd = WorldText[0].m_text.Split('.')[1].Split(' ')[0];
        string[] args = WorldText[0].m_text.Split(' ');
        EinherjarOpts.ParseEinherjarCommand(cmd, args, player ,talker, WorldText[0], cam, characters);

        WorldText[0].m_text = "";
    }
}
```

Hijacking world_text…eventually



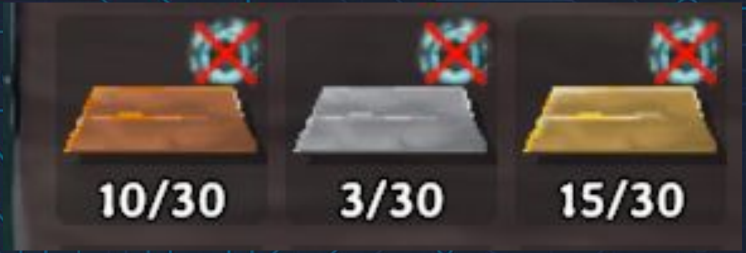
```
color=orange Grimmie /color :
color=#FFEB04FF I HAVE ARRIVED! /color
/color
color=orange Grimmie /color :
color=#FFFFFFFF [CHAT DEBUG] /w [text] -
Whisper
/s [text] - Shout
/die - Kill yourself
/resetspawn - Reset spawn point
/[emote]
Emotes: wave, sit, challenge, cheer, nonono,
thumbsup, point, blowkiss, bow, cower, cry,
despair, flex, comehere, headbang, kneel,
laugh, roar, shrug, dance, relax, toast, rest

color=orange Grimmie /color :
color=#FFEB04FF I HAVE ARRIVED! /color
color=orange Grimmie /color :
color=#FFFFFFFF [CHAT DEBUG] /w [text] -
Whisper
/s [text] - Shout
/die - Kill yourself
/resetspawn - Reset spawn point
/[emote]
Emotes: wave, sit, challenge, cheer, nonono,
thumbsup, point, blowkiss, bow, cower, cry,
despair, flex, comehere, headbang, kneel,
laugh, roar, shrug, dance, relax, toast, rest

color=orange Grimmie /color :
color=#FFEB04FF I HAVE ARRIVED! /color
/color
/color
/color
color=orange Grimmie /color :
color=#FFFFFFFF [CHAT DEBUG] /w [text] -
Whisper
/s [text] - Shout
/die - Kill yourself
/resetspawn - Reset spawn point
/[emote]
Emotes: wave, sit, challenge, cheer, nonono,
thumbsup, point, [illegible]ow, cower, cry,
despair, flex, comehere, headbang, kneel,
laugh, roar, shrug, dance, relax, toast, rest

color=orange Grimmie /color :
color=#FFEB04FF I HAVE ARRIVED! /color
color=orange Grimmie /color :
```

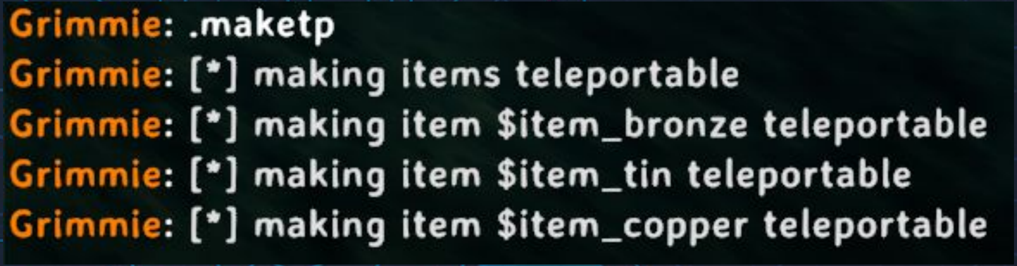
```
/die - Kill yourself
/resetspawn - Reset spawn point
/[emote]
Emotes: wave, sit, challenge, cheer, nonono, thumbsup,
point, blowkiss, bow, cower, cry, despair, flex, comehere,
headbang, kneel, laugh, roar, shrug, dance, relax, toast,
rest

 color=orange Grimmie /color :  color=#FFEB04FF I HAVE
ARRIVED! /color
 color=orange Grimmie /color :  color=#FFFFFFFF [CHAT
DEBUG] /w [text] - Whisper
/s [text] - Shout
/die - Kill yourself
/resetspawn - Reset spawn point
/[emote]
Emotes: wave, sit, challenge, cheer, nonono, thumbsup,
point, blowkiss, bow, cower, cry, despair, flex, comehere,
headbang, kneel, laugh, roar, shrug, dance, relax, toast,
rest

 color=orange Grimmie /color :  color=#FFEB04FF I HAVE
ARRIVED! /color
 /color
 color=orange Grimmie /color :  color=#FFFFFFFF [CHAT
DEBUG] /w [text] - Whisper
/s [text] - Shout
/die - Kill yourself
/resetspawn - Reset spawn point
/[emote]
Emotes: wave, sit, challenge, cheer, nonono, thumbsup,
point, blowkiss, bow, cower, cry, despair, flex, comehere,
headbang, kneel, laugh, roar, shrug, dance, relax, toast,
rest

 color=orange Grimmie /color :  color=#FFEB04FF I HAVE
ARRIVED! /color
 color=orange Grimmie /color :  color=#FFFFFFFF [CHAT
DEBUG] /w [text] - Whisper
/s [text] - Shout
/die - Kill yourself
/resetspawn - Reset spawn point
/[emot
```



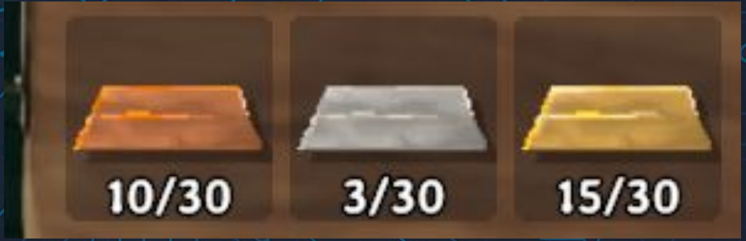
```
Grimmie: .help
Grimmie: help -  displays this menu -  .[command] [args]
Grimmie: entityesp -  activates entity esp
Grimmie: itemesp -  activates item esp
Grimmie: coords -  fetch player coordinates -  .coords
[player]
Grimmie: tp -  teleports player -  .tp [x] [z]
Grimmie: repair -  repairs all items
Grimmie: heal -  heals player -  .heal [player]
Grimmie: godmode -  toggle god mode (take no dmg past
1hp)
Grimmie: ghostmode -  toggle ghost mode (enemies ignore
you)
Grimmie: maketp -  makes all items in inventory
teleportable
Grimmie: searchitem -  search for item -  .searchitem
[itemname]
Grimmie: additem -  adds item to player inventory -  -
.additem [itemname]
Grimmie: wthr -  sets weather -  .wthr [weather]
Grimmie: players -  lists all connected players
Grimmie: plsmap -  does a funny
```

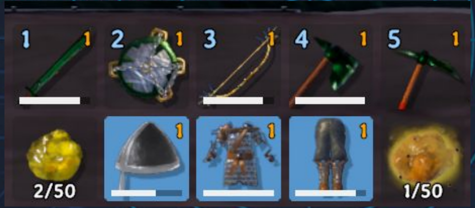
Free…From The Shackles Of A Workbench

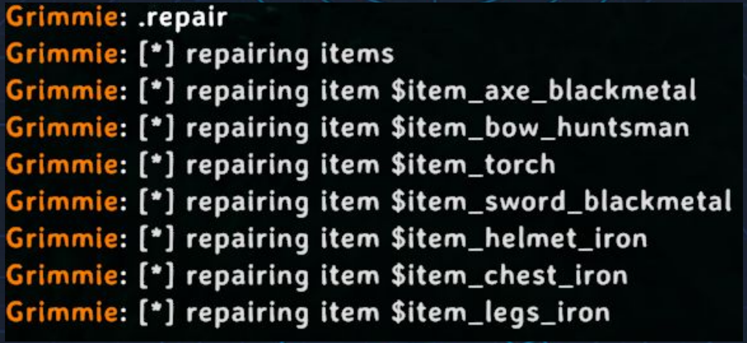




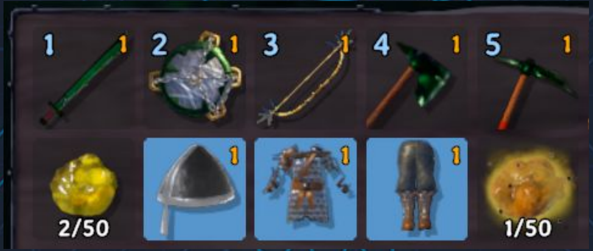
```
Grimmie: .maketp
Grimmie: [*] making items teleportable
Grimmie: [*] making item $item_bronze teleportable
Grimmie: [*] making item $item_tin teleportable
Grimmie: [*] making item $item_copper teleportable
```



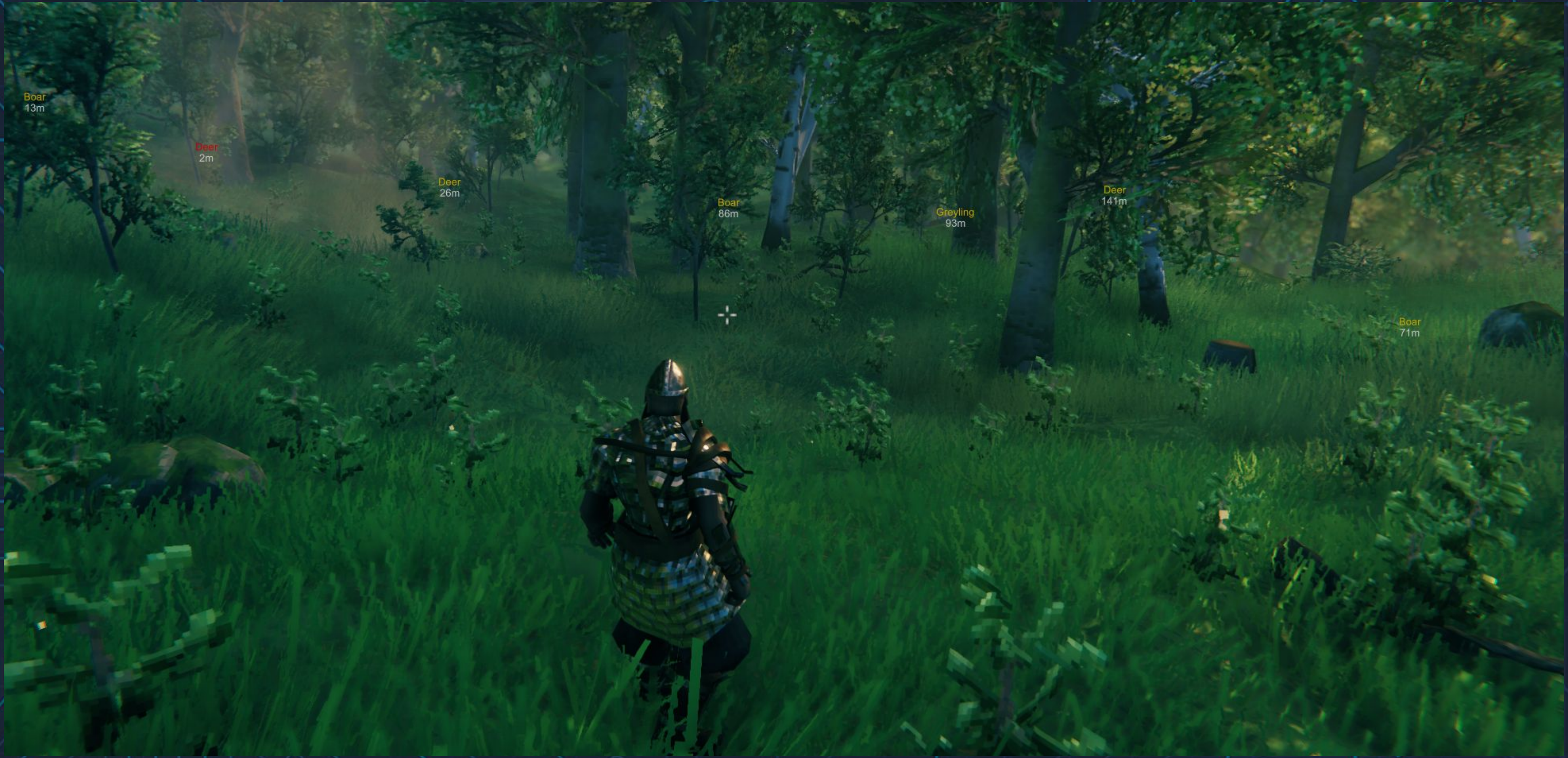




```
Grimmie: .repair
Grimmie: [*] repairing items
Grimmie: [*] repairing item $item_axe_blackmetal
Grimmie: [*] repairing item $item_bow_huntsman
Grimmie: [*] repairing item $item_torch
Grimmie: [*] repairing item $item_sword_blackmetal
Grimmie: [*] repairing item $item_helmet_iron
Grimmie: [*] repairing item $item_chest_iron
Grimmie: [*] repairing item $item_legs_iron
```

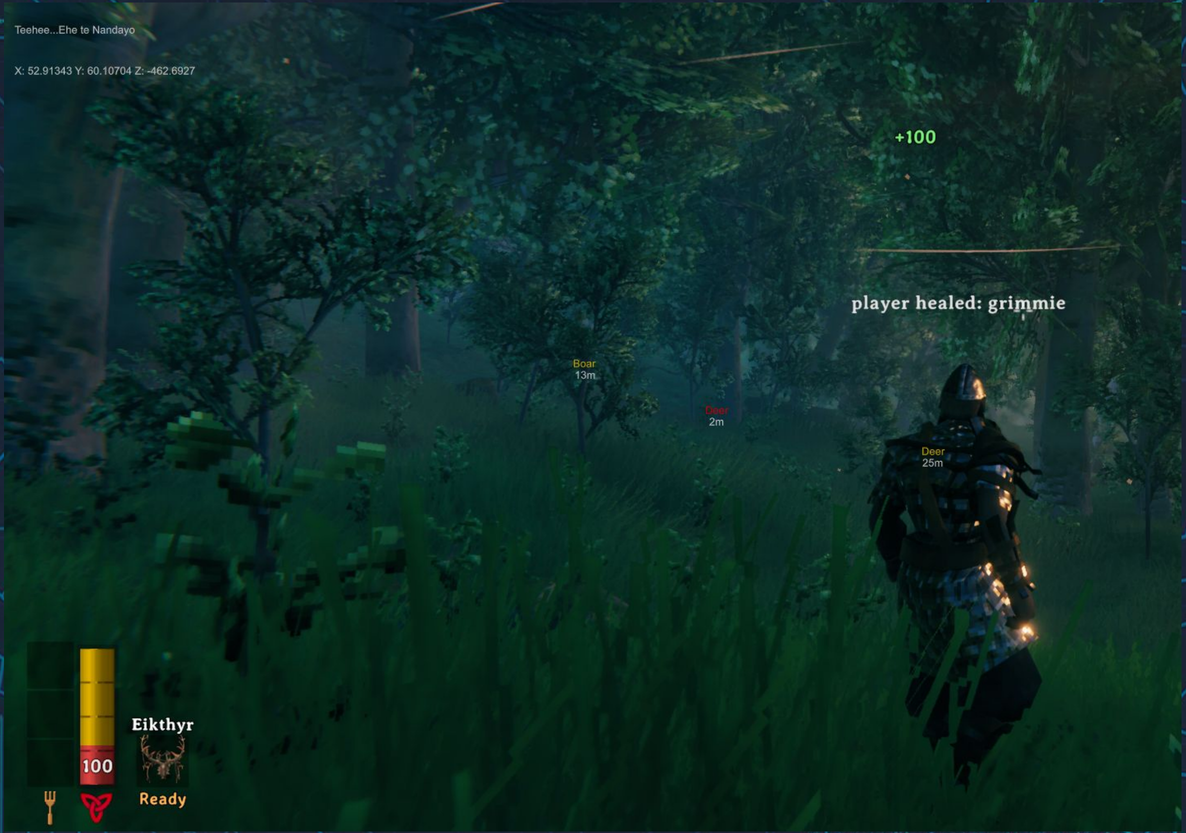


Oh Look, A Deer



> Loading Module: Einherjar.dll -> EntityESP()

Look Ma! NO FOOD!



Thanks & Resources

- Thanks for listening to me yap about Unity for a bit :)

- Resources
  - GuidedHacking.com - Paid, Free Stuff on Youtube
    - https://www.youtube.com/@GuidedHacking
  - GameHacking.gg - Free
  - Gamehacking.academy - Free
  - Game hacking site:github.com
    - https://github.com/dsasmblr/game-hacking
    - https://github.com/TheZong/Game-Hacking
    - https://github.com/AberFray/how-to-make-game-cheats-and-anticheats