Kode Forklaring

Indholdsfortegnelse

[Indexerings exemple 1](#_Toc505600049)

[1 Scripts 1](#_Toc505600050)

[1.1 Karakter og fjender 1](#_Toc505600051)

[1.1.1 Move 1](#_Toc505600052)

[1.1.2 Health 2](#_Toc505600053)

[1.1.3 Hitbox 2](#_Toc505600054)

[1.1.4 MovementAI 2](#_Toc505600055)

[1.1.5 TextScript 3](#_Toc505600056)

[1.2 Udstyr 3](#_Toc505600057)

[1.2.1 Bullet 3](#_Toc505600058)

[1.2.2 Weapon 3](#_Toc505600059)

[1.3 Ekstra 4](#_Toc505600060)

[1.3.1 ChildTriggerCollider 4](#_Toc505600061)

[2 Plugins 4](#_Toc505600062)

[2.1 Decals 4](#_Toc505600063)

[2.1.1 DecalObject 4](#_Toc505600064)

[2.1.2 BlitTexture2DExt 5](#_Toc505600065)

[2.1.3 DecalObjectEditor 5](#_Toc505600066)

[2.1.4 DecalWindow 5](#_Toc505600067)

[2.2 PrefabChanger 6](#_Toc505600068)

[2.2.1 PrefabChangerWindow 6](#_Toc505600069)

[2.2.2 PrefabOption 6](#_Toc505600070)

[2.2.3 ButtonInfo 6](#_Toc505600071)

[2.3 PrefabPlacer 7](#_Toc505600072)

[2.3.1 PrefabPlacerWindow 7](#_Toc505600073)

[2.4 MaterialReplacer 7](#_Toc505600074)

[2.4.1 MaterialReplacerWindow 7](#_Toc505600075)

[2.4.1.3.3 SetReplaceMaterial() 8](#_Toc505600076)

[2.5 RandomHouseColor 8](#_Toc505600077)

[2.5.1 RandomHouseColorParent 8](#_Toc505600078)

[2.5.2 RandomHouseColorEditor 8](#_Toc505600079)

# Indexerings exemple

Vores indexering at x er et tal. Bid mærke i at unity, public og private allrede har et bestemt tal.

**X Hoved Kategori**

**X.X Under Kategori**

**X.X.X Klasse Navn**

X.X.X.1.X Unity Events

X.X.X.2.X Public Method

X.X.X.3.X Private Method

# 1 Scripts

## 1.1 Karakter og fjender

### 1.1.1 Move

Denne klasse gør at vi kan bevæge os rundt med tastatur og kigge rundt med mus, den sørger også for at vi ikke kan dreje hovedet hele vejen rundt på x og z aksen. Vi kan også dukke os. Man kan fjerne musen på ’i’ og få den frem igen på ’i’.

#### 1.1.1.1.1 Start()

Vi gemmer nogle components og kameraet på gameobjectet.

#### 1.1.1.1.2Update()

Her er hvordan kontrol over gameobjectet bliver udført.

#### 1.1.1.3.1 CalculateMovement()

Får gameobjectet til at bevæge sig den retning man vil.

#### 1.1.1.3.2 RotateView()

Bevæger kameraet med musen.

#### 1.1.1.3.3 StartCrouching()

Gør at vi kan dukke os, når vi trykker på ’ctrl’.

#### 1.1.1.3.4 StopCrouching()

Gør at vi stopper med at dukke, når vi giver slip på ’ctrl’

#### 1.1.1.3.5 CanStand()

Sørger for at vi kun kan rejse os op når der ikke er noget oven over os. Vi laver max 4 raycast som tjekker hvert hjørne af CharacterController om der er noget over den.

#### 1.1.1.3.6 IsMouseLocked()

Her kan vi køre musen synlig eller usynlig.

### 1.1.2 Health

Giver et gameobject liv og mulighed for at kunne tage skade.

#### 1.1.2.1.1 Start()

Giver gameobjectet fuld liv og tildeler den metode der skal udføres når en hitbox bliver ramt Hvis der er tilføjet et template for healthbar vil den også blive oprettet her.

#### 1.1.2.1.2 Update()

Når gameobjectet dør vil vi gør den usynlig.

#### 1.1.2.2.1 TakeDamage()

Giver skade til dette gameobject. Denne metode bliver kaldt fra hitbox, hvis der bruges health bar vil den også blive opdateret her.

### 1.1.3 Hitbox

Den her klasse sørger for at, det kun er udvalgte layers som kan give skade til en collider. Man kan også vælge hvor meget % skade den skal tage.

#### 1.1.3.1.1 Start()

Her definere hvad der kan give skade til gameobjectet.

#### 1.1.3.2.1 OnGameObjectEnter()

Denne metode skal blive kaldt fra et andet script. Hvis vi bliver ramt af et gameobject som kan give os skade, tager vi skade.

### 1.1.4 MovementAI

Her inde har vi lavet en sphere collider som bruges med en trigger, triggeren bliver activeret når player kommer inden for dens række vide. Vi har lavet en metode som gør at man skal stå foran NPC'en før den begynder at følge efter playeren. Hvis der ikke er en player foran NPC'en og inden for dens sphere collider, går den rundt tilfældigt.

#### 1.1.4.1.1 Start()

Her gemmer vi components til senere brug. Så laver vi et nyt child gameobject med en SphereCollider og ChildTriggerCollider, så når en spillet kommer tæt på så ved vi det.

#### 1.1.4.1.2 Update()

Hvis vores gameobject ikke har nogen destination, så får den en ny tilfældig destination.

#### 1.1.4.3.1 TriggerOnStay()

Bliver kaldt fra et andet script. Tjekker om der er noget inden for dens trigger cirkel, hvis ja skal den følge efter det.

#### 1.1.4.3.2 RandomMove()

Har en random generator som får den til at gå forskellige steder hen.

#### 1.1.4.3.3 CanSeePlayer()

Sørger for at gameobject trigger kun virker foran den.

### 1.1.5 TextScript

I denne klasse henter vi HP og ammo fra Health og Weapon klasserne og gemmer dem i en string med noget tekst og udskriver dem i vores Ui/canvas.

#### 1.1.5.1.1 Start()

Henter Health og Weapon components fra parent, så vi ved hvor meget HP eller ammo vi har.

#### 1.1.5.1.2 Update()

laver 2 strings som vi så udskriver i en canvas.

## 1.2 Udstyr

### 1.2.1 Bullet

Denne klasse bruges til at flytte "Bullet"(gameobject). Når "Bullet"(gameobject) rammer et andet object kalder vi metoden OnGameObjectEnter() på det object vi ramte.

#### 1.2.1.1.1 Update()

Når gameobjectet er synlig flytter vi det hent mod endPoint hvis gameobjectet rammer noget på vejen kalder vi OnGameObjectEnter() på det object vi ramte Når gameobjecet rammer noget gør vi gameobjecet usynlig.

#### 1.2.1.1.2 OnTriggerEnter()

Tjekker om vi kalder OnGameObjectEnter på det object den vi ramte og gør gameobjectet usynlig bagefter.

#### 1.2.1.1.3 OnEnable()

Vi nulstiller bullet/gameobject når det bliver aktiveret.

#### 1.2.1.3.1 Fire()

Her beregner vi gameobject skal flyve hent til ud fra hvor spilleren kigger hen.

### 1.2.2 Weapon

I denne klasse gør vi at vi kan skyde og at vores skyd rammer rigtigt i forhold til vores kamera. Vi sørger også for at bruge de samme skyd som har været brugt eller lave nye hvis der ikke er flere at bruge, så vi ikke bruger ligeså meget resourse kraft. Det er også her vi reloader våbent når man trykker ’R’ eller der ikke er flere skyd tilbage.

#### 1.2.2.1.1 Start()

Vi henter kameraet, så vi ved hvor vi skal sende skydende hen. Giver spilleren fuld ammunition.

#### 1.2.2.1.2 Update()

Vi giver spilleren mulighed for at reloade og tvinger ham til at reloade når han løber tør.

#### 1.2.2.1.3 LateUpdate()

Affyre skyd på det rigtige tidspunkt. Hvis vi ikke udføre den i LateUpdate(), så kan vi ikke være sikre på at spillerens Rigidbody/krop er det rigtige sted

#### 1.2.2.3.1 GetBullet()

Hvis der er et bullet som er deaktiveret i spillet, bliver den genbrugt, og hvis der ikke er nogen deaktiveret bullets, bliver der lavet et nyt.

#### 1.2.2.3.2 Reload()

Efter 2 sekunder får spilleren fuld ammunition, eller det antal skyd de har tilbage.

## 1.3 Ekstra

### 1.3.1 ChildTriggerCollider

Når vores gameobject trigger et event vil den i stedet kalde en af vores properties f.eks. OnTriggerEnter() vil kalde propertie TriggerOnEnter. Dette script bliver tildelt til et child gameobject, så vi kan lave metoder inde i parent gameobject, som så kan blive sat på disse propertites.

#### 1.3.1.1.1 OnTriggerEnter()

Tjekker om den har fået tildelt en metode fra parent, hvis den har, vil metoden blive udført.

#### 1.3.1.1.2 OnTriggerStay()

Tjekker om den har fået tildelt en metode fra parent, hvis den har, vil metoden blive udført.

#### 1.3.1.1.3 OnTriggerExit()

Tjekker om den har fået tildelt en metode fra parent, hvis den har, vil metoden blive udført.

# 2 Plugins

## 2.1 Decals

### 2.1.1 DecalObject

Bruges ikke mere da vi fandt ud af at have for mange decals/gameobject fik spillet til at lagge vi har løst problemet ved at lave en shader i stedet for.

Dette script gør kun noget i editor mode og det den gør er at den laver en Quad hvor der bliver sat et sprite på. Dette script bruger også en CustomEditor (DecalObjectEditor.cs)

#### 2.1.1.1.1 OnEnable()

Opdatere decal når man starter Unity editor. Der er en fejl der gør at den også bliver kaldt Når man bare testet spillet

#### 2.1.1.2.1 BuildDecal()

Her laver vi gameobjectet om til en decal med billede. Der bliver ikke brugt shared material (kan være en grund til at det lagger) hvis den er tæt på et andet gameobject vil den tegne billedet på det andet gameobject (Der er et problem med størrelsesforholdet). Denne metode bliver kaldt fra et andet scripts(DecalWindow.cs, DecalObjectEditor.cs)

#### 2.1.1.3.1 CreateMesh()

Laver en firkantet mesh der gør det muligt at se gameobjectets materiale.

### 2.1.2 BlitTexture2DExt

Bruges ikke mere. Se DecalObject.cs beskrivelse.

Denne klasse er for extension metoder. Kombinere to Texture2D til en Texture2D. (Dette virker ikke helt som vi havde håbet da der er problemet med størrelsesforholdet.)

#### 2.1.2.2.1 Blit()

Her kombinere vi de to Texture2D der skal blive til et. Vi har hoved Texture2D(texture) også det andet(decal) der skal tilføjes på en angivet position(position).

### 2.1.3 DecalObjectEditor

Bruges ikke mere. Se DecalObject.cs beskrivelse

Vi har lavet dette CustomEditor så vi nemt kan flytte decal (CTRL + Klik) også kan vi se et billede af valgte decal plus vi kan ændre flere på en gang.

#### 2.1.3.1.1 OnEnable()

Når scriptet bliver aktiv, finder vi de properties der skal kunne blive ændret i og gemmer dem så vi ikke skal bruge resources på at finde dem hele tiden.

#### 2.1.3.1.2 OnSceneGUI()

Tjekker om man holder ctrl nede og klikker på en overflade, hvis man gør det flytter vi decal der hen.

#### 2.1.3.1.3 OnInspectorGUI()

Her ændrer vi udseende for DecalObject scripet i Inspector vinduet. Vi gør dette i et CustomEditor script så vi kan se billedet af decal.

### 2.1.4 DecalWindow

Bruges ikke mere. Se DecalObject.cs beskrivelse.

Et editor vindue der gør det nemt at oprette nye decals.

#### 2.1.4.1.1 OnGUI()

Udseende for vores vindue.

#### 2.1.4.2.1 ShowWindow()

Åbner vores vindue.

#### 2.1.4.3.1 Button\_CreateDecak\_Click()

Når man klikker på "Add Decal To Scene" knappen laver vi et nyt gameobject og giver den de Component den skal have og kalder .BuildDecal() fra DecalObject.cs

## 2.2 PrefabChanger

### 2.2.1 PrefabChangerWindow

Med dette kan vi sætte mange prefabs ind i en liste hvorefter vi så kan vælge en eller flere gameobjecter også sige at de skal skifte ud med det valgte prefab i listen.

#### 2.2.1.1.1 OnGUI()

Vores vindue udseende

#### 2.2.1.2.1 ShowWindow()

Åbner vores vindue.

#### 2.2.1.3.1 Button\_OnMakeChange()

Apply knap. Her udskifter vi de valgte gameobjecter.

#### 2.2.1.3.2 Button\_OnRemove()

Fjerner valgte option fra listen.

#### 2.2.1.3.3 InsertPrefabOption()

Tilføjer en ny option til listen.

### 2.2.2 PrefabOption

Option box der kan holde på en prefab med apply og remove knap. Har unik id så man kan kende forskel.

#### 2.2.2.2.1 PrefabOption()

Giver vores fields værdier.

#### 2.2.2.2.2 Display()

Udseende for option box.

### 2.2.3 ButtonInfo

En Button der kan udføre et event.

#### 2.2.3.2.1 ButtonInfo()

Giver vores fields værdier.

#### 2.2.3.2.2 Display()

Udseende for knap.

## 2.3 PrefabPlacer

### 2.3.1 PrefabPlacerWindow

Dette er et plugin til Unity som ikke har noget med spillet at gøre. Denne klasse gør at vi kan placere mange prefabs på en engang i stedet for at placer 1 af gangen. Når man holder musen nede bliver der placeret prefabs inden for den radius man har valgt. Man vælger selv prefabs.

#### 2.3.1. 1.1 OnGUI()

Tilføjer tingene der bliver vist i editor siden.

#### 2.3.1. 1.2 OnFocus()

Vi tilføjer et event til sceneview.

#### 2.3.1. 3.1OnSceneView()

Vi opretter en mulighed for at kunne fortryde(undo). Hvis man klikker på et object, begynder vi at placere prefabs.

#### 2.3.1. 3.2 RegisterUndo()

den her metode gør at vi kan lave undo(ctrl + z) og redo(ctrl + y).

#### 2.3.1. 3.3 HasCollider()

Tjekker om gameobjectet og dens child har en collider.

#### 2.3.1. 3.4 BeginInstantiateGameObjects()

Her placere vi de valgte prefabs i cirkler og hvis de er for tæt på hinanden bliver de sprunget over.

#### 2.3.1. 3.5 ValidatePrefabs()

Tjekker listen med prefabs igennem om de har colliders og at de ikke er null.

#### 2.3.1. 3.6 PlacePrefab()

Tjekker om prefabs kan blive placeret der hvor musen er.

#### 2.3.1. 3.7 GetPosition()

Tager center point og tilføjer radius og drejer den rundt om center point, med det angivet degree.

#### 2.3.1. 3.8 GetDegree()

Konvertere vores space variable om til grader så vi kan lave en cirkel.

#### 2.3.1. 3.9 GetBounds()

Finder størrelsen på det valgte gameobject og hvor meget vi kan tilføje for at den er over jorden.

#### 2.3.1. 3.10 ObjectFieldArray<T>()

Array version af EditorGUILayout.ObjectField.

## 2.4 MaterialReplacer

### 2.4.1 MaterialReplacerWindow

Den her klasse gør det muligt for os at udskifte materialer på mange forskellige objecter på samme tid og på en nem måde.

#### 2.4.1.1.1 OnGUI()

Her laver vi layout på vores vindue og fortæller hvad de forskellige ting skal gøre.

#### 2.4.1.2.1 ShowWindow()

Viser vinduet.

#### 2.4.1.2.2 ReplaceMaterial()

Leder efter et material og udskifter den med noget andet material. Man kan vælge at den også skal tjekke child igennem.

#### 2.4.1.3.1 GetMaterial()

Henter material fra gameobject.

#### 2.4.1.3.2 HasMaterial()

Tjekker om gameobjectet har et material.

### 2.4.1.3.3 SetReplaceMaterial()

Leder efter et material og udskifter den med noget andet material.

## 2.5 RandomHouseColor

### 2.5.1 RandomHouseColorParent

Klassen tilføjes til de gameobjecter som skal skifte. farve når man er i CustomEditor og klikker på en knap.

#### 2.5.1. 2.1 ChangeColorByRandom()

Giver gameobjectet en tilfældig farve fra vores liste af farver.

#### 2.5.1. 2.2 ReplaceMaterial()

Leder efter et material og udskifter den med noget andet material. Man kan vælge at den også skal tjekke child igennem.

#### 2.5.1. 3.1 GetMaterial()

Henter material fra gameobject.

#### 2.5.1. 3.2 HasMaterial()

Tjekker om gameobjectet har et material.

#### 2.5.1. 3.3 SetReplaceMaterial()

Leder efter et material og udskifter den med noget andet material.

### 2.5.2 RandomHouseColorEditor

Tilføjer en knap til gameobject som har RandomHouseColorParent component, når man trykke på knappen skifter den farverne.

#### 2.5.2.1.1 OnInsectorGUI()

Tilføjer en knap, når man trykker på den skifter den farverne.