Player

The Player object is a humanoid character that is controlled by the user and is capable of jumping, climbing staircases, climbing ladders and crouching into tunnels. The player can use the melee and the shooter module to attack the enemy and has stats like health and armor[1]. The player receives damage either by being hit with a melee weapon, by being shot, or falling to his death.

# How it works

Coming Soon.

# Variables

## Local Variables

These variables are found on the Player object itself:

* 3rdPersonMotor\*
* 1stPersonMotor\*
* 1stPersonMotorOriginal(?)\*
* tunnelMotor\*
* activeCamera\*
* target
* location
* running
* allowRunning
* crouching
* allowCrouching
* allowJumping
* blocking
* onLadder
* inTunnel
* pickUpWeapon
* showItemUI
* itemUIName
* itemUICurrentName
* pickUpWeaponName
* isAiming
* hasFists1
* hasSword1
* hasGun1
* hasGun2
* hasGun3
* equippedFists1
* equippedSword1
* equippedGun1
* equippedGun2
* equippedGun3

These variables are found on the Variables object (they are here and not on the player object because they don’t need to be accessed outside of the player):

* motorUsed(?)
* shootTimer
* respawnItems
* paused

\*I’d like to move these to the Variables object.

## RespawnList

Empty on start.

## PlatformList(?)

Empty on start.

## FocusList(?)

Empty on start.

# Triggers

## Start

Trigger: On Start

Calls action SetVariables

Calls action SetAttributes

Calls action SaveLocation

Calls action AddStamina

Calls action DrawFists1

## onLadder

Trigger: On variable change (Player(onLadder))

Calls conditions onLadderCheck

## inTunnel

Trigger: On variable change (Player(inTunnel))

Calls conditions inTunnelCheck

## PauseMenu

Trigger: On Key Down(Joystick 7, Escape)

Calls conditions PauseMenuCheck

## View

Trigger: On Key Down(Joystick 6, V)

Calls conditions ViewCheck

## Jump

Trigger: On Key Down(Joystick 0, Space)

Calls conditions JumpCheck

## Crouch

Trigger: On Key Down(Joystick 9, Left CTRL)

Calls conditions CrouchCheck

## Running

Trigger: On Key Down(Joystick 8, Left Shift)

Calls conditions RunningCheck

Trigger: On Key Up(Joystick 8, Left Shift)

Calls conditions DontRun

## ActuallyRunning

Trigger: On Key Down(?,W)

Calls conditions ActuallyRunningCheck

Trigger: On Key Up(?,W)

Calls conditions ActuallyRunningCheck

## OnStep

Trigger: On Step (Player, any)

Call conditions onStepCheck

## LetGo

Trigger: On Key Down(Joystick 3, E)

Calls conditions LetGoCheck

# Conditions

## onLadderCheck

If onLadder = true

And activeCamera = 1stPersonMotor

Execute actions DisableSwitch1stPerson

If onLadder = true

And activeCamera = 3rdPersonMotor

Execute actions DisableSwitch

Else execute actions EnableSwitch

## inTunnelCheck

If inTunnel = true

And activeCamera = 3rdPersonMotor

Execute actions ActivateTunnelMotor

## PauseMenuCheck

Follow pattern (true,false)

Execute actions Pause/Execute actions Unpause

## ViewCheck

Follow pattern (true, false)

Execute actions 1stPerson/Execute actions 3rdPerson

## JumpCheck

If canJump = true

Execute actions Jump

## CrouchCheck

If crouching = false

And canCrouch = true

Execute actions Crouch

If crouching = false

And canCrouch = false

Execute actions Crouch

If crouching = true

And canCrouch = false

Do nothing

Else execute actions DontCrouch

## CrouchStateCheck

If activeCamera = 1stPersonMotor

Execute Crouch1stPerson

Else execute Crouch3rdPerson

## DontCrouchStateCheck

If activeCamera = 1stPersonMotor

Execute DontCrouch1stPerson

Else execute DontCrouch3rdPerson

## RunningCheck

If Player Stamina >= 1

And crouching = false

And allowRun = true

Execute actions Run

Else execute actions StopRunning

## ActuallyRunningCheck

If Player Stamina >= 1

And CanRun = true

And W is being pressed

Execute actions IncreaseFOV

If Player Stamina <= 1

And CanRun = true

And W is being pressed

Execute actions DecreaseFOV

Else execute actions DecreaseFOV

## OnStepCheck

If isAiming = true none,

If Player Stamina >= 1

If Player CanRun

Execute actions SubtractStamina

If Player Stamina <= 1

Execute actions StopRunning

## LetGoCheck

If onLadder = true

Execute actions LetGo

## MotorUsedCheck

If motorUsed = 1stPersonMotor

Execute actions ResetMotor

# Actions

## SetVariables

Wait 0.01s

Local variable target is none(?)

Gather RespawnPlayer to RespawnList

Global variable player is Player

Lock cursor, set invisible

Local variable 1stPersonMotorOriginal is 1stPersonMotor(?)

## SetAttributes

Wait 0.1s

Set Player Health to 100

## SaveLocation

Local variable location is Vector3(Player)

Restart actions

## AddStamina

Add 5 Stamina to Player

Wait 1s

Restart actions

## SubtractStamina

Subtract 5 Stamina from Player

## Pause

Set active PauseMenu

paused = true

Timescale = 0

## Unpause

Set inactive PauseMenu

paused = false

Timescale = 1

## #1stPerson

A whole bunch of stuff.

## #3rdPerson

A whole bunch of stuff.

## Jump

Jump Player

## Crouch

Call conditions CrouchStateCheck

Change Player Height (0.9)

Player SetRunSpeed (4.5)

crouching = true

allowJumping = false

Change 1stPersonMotor settings (0.0, 1.1, 0.15)

Change 3rdPersonMotor settings (0.25, 1.1, 0)

## DontCrouch

Call conditions DontCrouchStateCheck

Change Player Height (2)

Player SetRunSpeed (9)

crouching = false

allowJumping = true

Change 1stPersonMotor settings (0.0, 1.55, 0.15) // And headbobbing?

Change 3rdPersonMotor settings (0.25, 1.7, 0)

## Run

Player CanRun = true

running = true

Call conditions ActuallyRunningCheck

## StopRunning

Player CanRun = false

running = false

Call conditions ActuallyRunningCheck

## IncreaseFOV

Change FOV to 95 (0.25)

## DecreaseFOV

Change FOV to 90 (0.25)

## LetGo

Player let go of climbable

## DisableSwitch1stPerson

motorUsed = 1stPersonMotor

Execute 3rdPerson

Execute DisableSwitch

## DisableSwitch

Set inactive View

Set inactive Crouch

Set inactive SwitchWeapon

Set inactive Crosshair

## EnableSwitch

Call MotorUsedCheck (wait)

Set active View

Set active Crouch

Set active SwitchWeapon

Call CrosshairCheck (wait)

## ActivateTunnelMotor

Change camera to TunnelMotor (0.25)

## Crouch1stPerson

Change Player state (CrouchFP)

## Crouch3rdPerson

Change Player state (Crouch)

## DontCrouch1stPerson

Change Player state (LocomotionFP)

## DontCrouch3rdPerson

Change Player state (Idle)

## ResetMotor

Execute 1stPerson

motorUsed = none

WeaponSystem

This is such a big and complicated part of the player, I felt like it needed its own section. Hopefully this way I can reuse it on different players, if I need to, while enabling me to edit the WeaponSystem just once instead of on every player.

# Triggers

## healthDecrease

Trigger: On Attribute Change(Player(Health, decrease))

Calls conditions HealthDecreaseCheck

## showItemUI

Trigger: On Variable Change(Player(showItemUI))

Calls conditions ItemUICheck

## pickUpWeapon

Trigger: On Variable Change(Player(pickUpWeapon))

Calls conditions WeaponLimitCheck

## SwitchWeapon

Trigger: On Key Down (Q)

Calls conditions SwitchWeaponCheck

Calls conditions RemoveWeaponCheck

## Melee

Trigger: On Key Down(Joystick 1, F)

Calls conditions MeleeCheck

Trigger (MeleeExtraTrigger): OnAxisRawDown(RT, positive), On Mouse Down(Left)

Calls conditions MeleeCheck

## Blocking

Trigger: On Axis Raw Down(LT, positive), On Mouse Down(Right)

Calls conditions BlockingCheck

Trigger: On Axis Raw Down(LT, negative), On Mouse Up(Right)

Calls conditions StopBlockingCheck

## Aim

Trigger: On Axis Raw Down(LT, positive), On Mouse Down(Right)

Calls conditions AimCheck

Trigger: On Axis Raw Down(LT, negative), On Mouse Up(Right)

Calls conditions StopAimCheck

## Shoot

Trigger: On Axis Raw Down(RT, positive), On Mouse Down(Left)

Calls conditions ShootCheck

## Reload

Trigger: On Key Down(Joystick 2), On Key Down(R)

Calls conditions ReloadCheck

# Conditions

## WeaponLimitCheck

If pickUpWeapon = true

And WeaponDrawList < 2

Execute actions KeepWeapon

If pickUpWeapon = true

And WeaponDrawList = 2

Execute actions RemoveWeapon

## PickUpCheck

Example Sword1:

If pickUpWeaponName = Sword1

And equippedSword1 = false

Execute actions PickUpSword1

## SwitchWeaponCheck

If WeaponDrawList < 2, else execute actions SwitchWeapon

## RemoveWeaponCheck

If WeaponRemoveList < 2, else execute actions SwitchRemoveWeapon

## HotBarCheck

Example Sword1:

If equippedSword1 = true

Execute actions PrimarySword1

## HotBarSecondaryCheck

Example Sword1:

If hasSword = true

And equippedSword1 = false

Execute actions SecondarySword1

## BlockingCheck

If Player is Attacking

Then execute RetryBlockingCheck

If Player is not Attacking

And Player is armed (melee)

And Player is unarmed (shooter)

And Player is not blocking

Then execute Block

## StopBlockingCheck

If Player is armed (melee)

And Player is blocking

Then execute DontBlock

## GetHitCheck

If Player is blocking

And Player Health >= 1

Then nothing (yet)

If Player is not blocking

And Player Health >= 1

Execute GetHit

## HolsterSheatheCheck

If Player is armed (shooter)

Execute actions Holster

If Player is armed (melee)

Execute actions Sheathe

## ResumeAimingCheck

PC: If Mouse Right being pressed

And isAiming = false

Then execute CallAimCheck

Xbox: If LTInUse = true

And isAiming = false

Then execute CallAimCheck

## ResumeRunningCheck

PC: If Left Shift being pressed

And running = false

Then execute CallRunCheck

Xbox: If Joystick 8 being pressed

And running = false

Then execute CallRunCheck

## ResumeBlockingCheck

PC: If Mouse Right being pressed

And blocking = false

Then execute CallBlockingCheck

Xbox: If LTInUse = true

And blocking = false

Then execute CallBlockingCheck

## ItemUICheck

If showItemUI = true

And WeaponDrawList < 2

Execute ShowPickup

If showItemUI = true

And WeaponDrawList = 2

Execute ShowReplace

Else execute DontShow

## MeleeCheck

If Player is armed (melee)

Then execute Melee

## ShootCheck

If Player is armed (shooter)

And isAiming = true

Then execute ShootAimed

If Player is armed (shooter)

And isAiming = false

Then execute ShootUnaimed

## ArmCheck

If activeCamera = 1stPersonMotor

Then execute MoveArms

## AimTimerCheck

If aimTimer = 0

Execute DontAim

## AimCheck

If Player is armed (shooter)

And equippedGun3 = true

And crouching = false

Then execute AimSniper\*

If Player is armed (shooter)

And isAiming = false

Then execute Aim

\*I’ve left this here so I can implement a sniper later, by default it is in the first person package I used but now it doesn’t work because Gun3 does not exist yet.

## DontAimCheck

If Player is armed (shooter)

And isAiming = true

Execute DontAim

## AimStateCheck

If crouching = true

And isAiming = true

Execute Crouch

If crouching = true

And isAiming = false

Execute Crouch

If crouching = false

And isAiming = true

Execute AimState

If crouching = false

And isAiming = false

Execute ResetState

## ArmsCheck

If activeCamera = 1stPerson

And isAiming = true

Execute ArmsMove

If activeCamera = 1stPerson

And isAiming = false

Execute ArmsReset

## ReloadCheck

If Player is armed (shooter)

Execute Reload

## HealthDecreaseCheck

If Player Health <= 0

Execute Death

## DropCheck

Example Sword1:

If dropItems = true

And currentWeaponName = “sword1”

Execute DropSword1

# Actions

## KeepWeapon

Call conditions PickUpCheck

## RemoveWeapon

Execute WeaponRemoveList(1)

Remove WeaponDrawList(1)

Remove WeaponRemoveList(1)

Call conditions PickUpCheck

## SwitchWeapon/SwitchRemoveWeapon

Ngl, I do some hocus pocus to switch the lists around. There is most likely a better way, but it works fine. And then, on SwitchWeapon only, I execute the first actions on the WeaponDrawList.

## ClearHotBar

Set inactive Sword1 (the primary and the secondary icons in the hotbar)

repeat for other weapons

Set false equippedSword1

repeat for other weapons

## Pickup

Example PickUpSword1:

Add DrawSword1 to WeaponDrawList

Add RemoveSword1 to WeaponRemoveList

Execute Actions WeaponDrawList(1)

Set pickUpWeapon to false

Clear pickUpWeaponName

## Draw

Example DrawSword1:

Execute actions ClearHotBar (wait)

Set equippedSword1 to true

Call conditions HotBarCheck (wait)

Call conditions HolsterSheatheCheck (wait)

Call conditions HolsterSheatheCheck (wait)

itemUICurrentName = “sword”

Execute actions StopRunning (wait)

Execute actions StopAim (wait)

Execute actions StopBlock (wait)

Player draw Sword1

Wait 0.01s

Set inactive Crosshair

Set active MeleeExtraTrigger

Call conditions ResumeRunningCheck (wait)

Call conditions ResumeBlockingCheck (wait)

To draw a gun is a little bit different; you have to draw the gun instead of the sword, add ammo to the gun,draw a melee weapon so you can use the gun for melee (optional), and instead of doing the ResumeBlockingCheck, do a ResumeAimingCheck since you can’t block, but you can aim with a gun. I do this because before drawing something you need to be completely stopped, not aiming, not blocking, to avoid glitches. And after drawing you expect to continue running, aiming or blocking if you are pressing those buttons. There is also DrawFists1, which is executed at the start of the game, and when you equip guns. This action just draws Fists1, sets hasFists1 to true and sets equippedFists1 to true. With this equipped you can punch, with or without holding a gun, but not with a sword.

## Remove

Example RemoveSword1:

hasSword1 = false

Call DropCheck

## Drop

Example DropSword1:

Instantiate Sword1Item

## HotBar

Example PrimarySword1:

Set active Sword1

Call HotBarSecondaryCheck

Example SecondarySword1:

Set active Sword1

## Holster

Player holster weapon

## Sheathe

Player sheathe weapon

## CallAimCheck

Call AimCheck\*

\*This one, CallRunningCheck and CallBlockingCheck work the same and are very simple. They are used on ResumeAimCheck etc. Maybe I could simplify it but that does not matter so much right now.

## GetHit

Player do gesture FrontHit4

Player Health -25.1

## Block

Player start blocking

Set RunSpeed 0

blocking = true

## RetryBlockingCheck

Wait 0.01s

Call BlockingCheck

## Melee

Call StopBlockingCheck

Input Player Melee A

Call ResumeBlockingCheck

## Death

A BUNCH OF STUFF

## Reload

Reload Player weapon

## ToggleLT

Toggle LTInUse

## ShowPickup/ShowReplace

Set active Pickup/Replace

## DontShowItemUI

Set inactive Pickup

Set inactive Replace

## ShootAimed

isShooting = true

Player shoot

isShooting = false

## ShootUnaimed

isShooting = true

Cancel AimTimer

Player aim camera direction

Wait 0.01s(?)

Call ArmCheck

Player shoot

aimTimer = 1

Execute AimTimer

## AimTimer

aimTimer - 1

Wait 1s

Call AimTimerCheck (wait)

Restart action

## Aim

Wait 0s(?)

Cancel AimTimer

allowRunning = false

isAiming = true

Call RunningCheck

Call AimStateCheck

Call ArmCheck

Player aim camera direction

## StopAim

Cancel AimTimer

Wait 0s(?)

allowRunning = true

isAiming = false

isShooting = false

Player stop aiming

Call ResumeRunningCheck

Call AimStateCheck

Call ArmCheck

## ArmsMove

Change Left Arm parent LeftArm

Change Right Arm parent RightArm

## ArmsReset

Change Left Arm parent Left shoulder

Change Right Arm parent Right shoulder

## AimState

Player state AimState

## ResetState

Player reset state

Enemy

The Enemy object is a humanoid character capable of using the navmesh, including off-mesh links that allow it to jump off platforms, onto staircases, climb ladders, smash vents and crouch into tunnels. The enemy has stats like health and armor[1], behavior and perception so it can see and walk and can use the melee and the shooter module to attack the player. The enemy will idle until it sees the player, after it will try to attack the player. The player can choose to run away and hide after which the enemy will try to look for the player or return to idle. Ofcourse, the player can also attack the enemy. The enemy receives damage based on weapons used or impact location, for example shooting the enemy in the head kills it instantly.

# How it works

A lot of things happen on start.

SetVariables is called, which sets the variables to be used later.

Then, SetAttributes is called, which sets the attributes of the enemy to their default.

After that DrawWeapons is called, making the enemy draw its gun and equipping its fists.

Then, GatherTargets starts, which repeats indefinitely.

And at last RotateUI is called which makes the healthbar of the enemy rotate towards the player constantly.

Ofcourse, it would be better if a lot of the things mentioned above were variables, but I have not made that yet. And from the looks of it it seems like some of these won’t be able to be turned into variables so maybe it is better (and easier) to just edit everything by hand when making a new enemy character. Still, it would be nice if you could do this from one local variable component, instead of going through all these actions.

Then, the enemy has a couple triggers. These are placed on the enemy itself, and on its body parts, and when they are triggered the enemy can receive damage or die completely. There is also a trigger which checks to see if the enemy needs to crouch or not to force it to crouch or stand up.

The rest is all done by using the behavior module. If the enemy cannot see the player, it will call the actions Idle which makes the enemy wander about the level based on markers placed on the level. Once it sees the player, the enemy will start following the player, until it is close enough after which it will start to shoot at the player. If the player chooses to run away and hide, the enemy will try and look for the player, and depending on the outcome of this search the enemy will resume attacking, or resume idling.

The last thing to explain would be how the enemy uses the navmesh. There is an action on the enemy object itself (at least for now[3]), which makes the enemy do a jump animation. This action is called for all downward jumps generated by the navmesh, using the script *AgentLinkMover*. For all upward “jumps” (using that word loosely here) I had to use manual offmeshlinks, because these are not generated by the navmesh. These jumps are called PlatformPoints and are placed near surfaces the enemy should be able to jump on, for example the side of staircases or low walls. This way the enemy movement is much smoother and more like the player’s. These jumps also enable the enemy to climb ladders (and attack others on ladders), and crouch into tunnels (and smash tunnels open). Depending on the actions on these offmeshlinks the enemy does different animations while it is traveling on the offmeshlink. There is another script on the enemy called *LinkCompleter*, which has an action called isOnOffMeshLink that is sometimes used to check whether or not the enemy is currently traveling on an offmeshlink. This is because performing actions during this traveling should not be done and can break its behavior.

All these things put together make the enemy able to walk around the level, jump on platforms, climb ladders, crouch into tunnels, find the player, shoot at or punch the player, and eventually die by getting shot at or punched by the player.

# Variables

## Local Variables

These variables are found on the Enemy object itself.

* target
* targetLocation
* isOnOffMeshLink
* canCrouch
* crouching
* needsToCrouch
* attackerWeapon

## MarkerList

Empty on start.

## RespawnList

Empty on start.

# Triggers

## OnTagEnter(Melee)\*

Stores collider as local variable attackerWeapon

Calls conditions BlockingCheck

## OnTagEnter(Sword)\*

Stores collider as local variable attackerWeapon

Calls conditions BlockingCheck

## Start

Trigger: On Start

Calls action SetVariables

Calls action SetAttributes

Calls action DrawWeapons

Calls action GatherTargets

Calls action RotateUI

## healthDecrease

Trigger: On Attribute Change(OnDecrease(Enemy[Health]))

Calls conditions HealthCheck

## canCrouch

Trigger: On Variable Change(Enemy[canCrouch])

Calls conditions CrouchCheck

*\*These triggers are set on the Enemy object itself, rather than separate triggers.*

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# Conditions

## HealthCheck

If Enemy[Health] <= 0

Execute action Death

## BlockingCheck

If Enemy is blocking

And Enemy[Health] >= 1

Execute action Blocking

If Enemy is not blocking

And Enemy[Health] >= 1

And attackerWeapon has tag Melee

Execute action GetHitFists

If Enemy is not blocking

And Enemy[Health] >= 1

And attackerWeapon has tag Sword

Execute action GetHitSword

## CrouchCheck

If crouching is false

And canCrouch is true

And needsToCrouch is true

Execute action Crouch

If crouching is false

And canCrouch is false

And needsToCrouch is true

Execute action Crouch

If crouching is true

And canCrouch is false

And needsToCrouch is true

Do nothing

Else execute actions DontCrouch

## DamageAnimCheck

If crouching is false

Execute action DamageAnim

Else execute action DamageAnimCrouch

## NavMeshCheck

If isOnOffMeshLink is true

Execute action RetryNavMeshCheck

# 

# 

# 

# 

# 

# 

# 

# 

# 

# Actions

## SetVariables

Wait 0.01s

Gather with tag RespawnEnemy into MarkerSpawnList

Gather with tag Marker into MarkerList

Gather with tag Player into Enemy[2]

## SetAttributes

Set Health of Enemy to 100

## DrawWeapons

Draw Gun1 with Ammo1NPC

Wait 0.1s

AddToMagazine 6 Ammo1NPC

Draw FistsEnemy

## GatherTargets

Gather with tag Player from Enemy(1000) into local variable target

Restart action

## RotateUI

GameObject UI look at Player

Restart Action

## Blocking

*Nothing yet.*

## DamageFists

Subtract 26 Health from Enemy

Call conditions DamageAnimCheck

## DamageSword

Subtract 51 Health from Enemy

Call conditions DamageAnimCheck

## DamageBullet

Subtract 26 Health from Enemy

Call conditions DamageAnimCheck

## DamageHead

Subtract 100 Health from Enemy

Call conditions DamageAnimCheck

## DamageAnim

Do gesture Character@frontHit4 and wait

Look at target

## DamageAnimCrouch

Do gesture Character@frontHit4(upperbody) and wait

Look at target

## Death

Stop behavior Enemy

Cancel Idle

Cancel Melee

Cancel LookForTarget

Cancel DamageAnimation

Stop Follow

Let Go of Climbable

Reset state

Stop Gesture

Ragdoll

playerScore + 1

Wait 3s

Set inactive UI

Call conditions RespawnList[first]

## Crouch

Call method linkCompleter.isOnOffMeshLink

Call conditions NavMeshCheck

Wait 0s *(?)*

Change state to Crouch

Set Can Run to false

Set crouching to true

## DontCrouch

Call method linkCompleter.isOnOffMeshLink

Call conditions NavMeshCheck

Wait 0s *(?)*

Reset state

Set Can Run to true

Set crouching to false

## RetryNavMeshCheck

Call method linkCompleter.isOnOffMeshLink

Wait 0.1s

Call conditions NavMeshCheck

# 

# 

# 

# 

# 

# 

# 

# 

# Behavior

## Idle

Call method linkCompleter.isOnOffMeshLink

Call conditions NavMeshCheck

Wait 0.1s

Stop aiming

Stop follow

Move to MarkerList[Random] and wait

Clear MarkerList

Gather with tag Marker into MarkerList

## Melee

Input Melee A

Wait between 0.3 and 1s

## Shooter

Aim at target

Wait between 0.5 and 2s

Single shoot

## LookForTarget

Call method linkCompleter.isOnOffMeshLink

Call conditions NavMeshCheck

Wait 0.1s

Stop Follow

Rotate towards target

Set variable targetLocation to variable target/location

Move to targetLocation

# 

# 

# Notes

[1] Not using armor for anything right now.

[2] It’s been a while since I made this, I hope to make this it’s own separate list in the future but for now I keep it this way because it works.

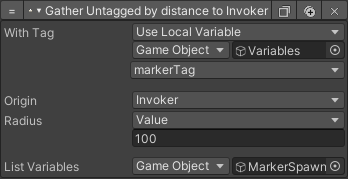
[3] It’s been a while since I made this, and it is very complicated. I keep it this way for now but if I ever need to change anything I will try and make this better too.

Spawner

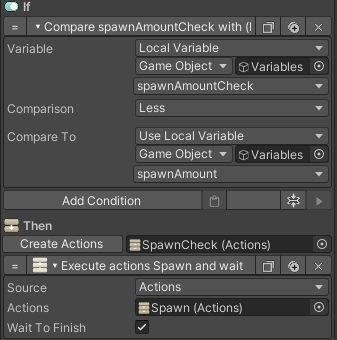
This object can spawn any amount of characters, including player characters. The character(s) will spawn on any marker with the right tag. It is possible to respawn the character(s) one by one, as a mob, or to have no respawning at all. It is possible to set a delay before respawning. It is possible to set the amount of bodies that stay on the scene before they despawn, despawn all bodies immediately, despawn the bodies as a mob, or to not despawn them at all.

# How it works

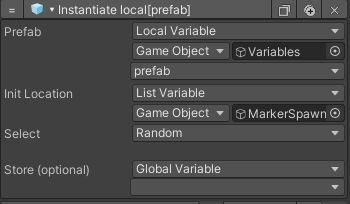
On start, SetVariables is called, which sets the variables to be used later. All markers with the tag matching the local variable *markerTag* are gathered into the MarkerSpawnList.



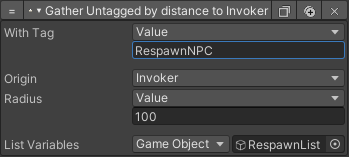
SpawnCheck is also called on start. This condition will call Spawn as many times as the amount set to the local variable *spawnAmount*.



The Spawn action will instantiate a character prefab, which is set to the local variable *prefab*, at one of the markers in the MarkerSpawnList chosen at random.



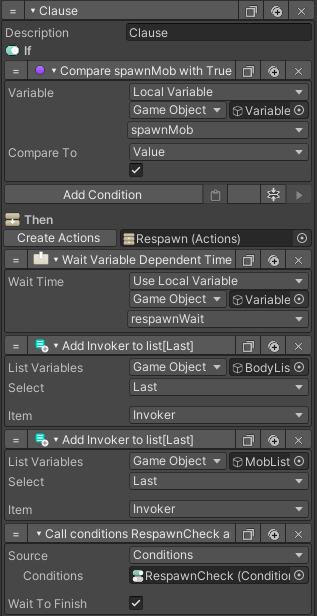
If respawning is necessary, the character prefab should have an action that is called on start, which gathers the Respawn condition which should then be called when the character dies. The value set here should match the tag of the Respawn condition.



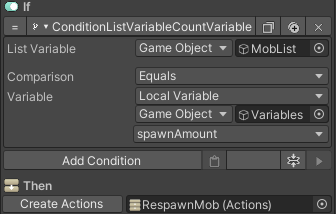
In this case Respawn has the tag RespawnNPC.



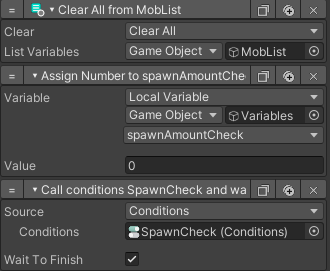
When this condition is called it will check if the local variable *spawnMob* is set to true. If so, it will wait for the amount of time set to the local variable *respawnWait* before adding the body to the BodyList and to the MobList. Then the RespawnCheck is performed.



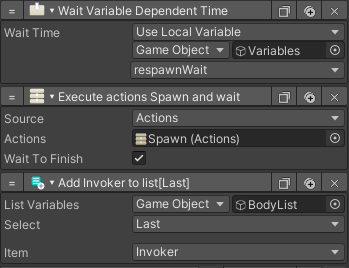
The RespawnCheck will compare the amount of bodies in the MobList to the original *spawnAmount*. This uses a custom action called *ListVariableCountKrabba*. If the amount of bodies matches the *spawnAmount* it will call RespawnMob.



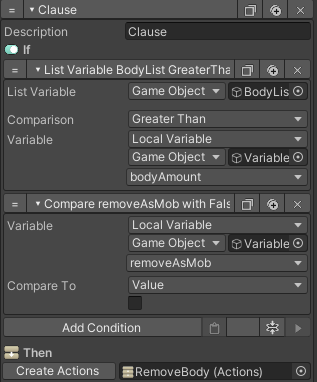
RespawnMob will clear the MobList and then reset and perform the SpawnCheck again.



If instead *spawnMob* is not set to true, Respawn will call the action RespawnSingle. This action also uses the optional *respawnWait* time before calling Spawn and adding the body to the BodyList.

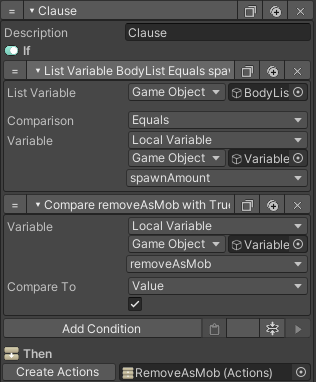


Each time a player’s body is added to the BodyList, a condition is called by the trigger TriggerBodyList. The first clause checks whether the amount of bodies in the BodyList is more than the number set in the local variable *bodyAmount*, and if *removeAsMob* is set to false. If so,it will call RemoveBody.



RemoveBody will destroy the first body in the BodyList, then call the conditions again and this will repeat until enough bodies have been destroyed.

The second clause checks whether the amount of bodies in the Body list equals the number set in the local variable *spawnAmount*, and if *removeAsMob* is set to true. If so, it will call RemoveAsMob.



RemoveAsMob will destroy the first body in the BodyList, then call the conditions again and this will repeat until all bodies have been destroyed.

# Actions

## SetVariables

Wait 0.01s

Gather with markerTag into MarkerSpawnList

## SpawnCheck

*If* spawnAmountCheck < spawnAmount

*Then* Execute Spawn

spawnAmountCheck + 1

Call SpawnCheck

## Spawn

Wait 0.01s

Instantiate prefab, location MarkerSpawnList[Random]

## Respawn

*If* spawnMob = True

*Then* Add Invoker to BodyList[Last]

Add Invoker to MobList[Last]

Call RespawnCheck

*Else* Call RespawnSingle

## RespawnCheck

If MobList = spawnAmount

Call RespawnMob

## RespawnMob

Wait(respawnWait)

Clear MobList

spawnAmountCheck = 0

Call SpawnCheck

## RespawnSingle

Wait(respawnWait)

Execute Spawn

Add Invoker to BodyList[Last]

## RemoveBody

Wait(0.01)

Destroy BodyList[First]

Remove BodyList[First]

Call conditions TriggerBodyList

# Triggers

## Start

Trigger: On Start

Calls action SetVariables

Calls action SpawnCheck

## TriggerBodyList

Trigger: On Variable Change(BodyList[Last])

Calls conditions:

*If* BodyList > BodyAmount

*Then* Call RemoveBody

# Variables

## Local Variables

markerTag

type: String

value: Use the name of the tag used on the markers where you want the character to spawn

spawnAmount

type: Number

value: Set to the amount of characters you want to spawn.

spawnMob

type: Bool

value: Set to true if you want the characters to spawn as a mob, set to false if you want them to spawn as individuals.

respawnTag

type: String

value: Use the name of the tag used on the Respawn action for this spawner.

respawnWait

type: Number

value: Set to the amount of time you want to wait before respawning characters.

bodyAmount

type: Number

value: Set to the amount of bodies you want to leave on the scene before despawning them

prefab

type: GameObject

value: Set to the prefab character you want to spawn

spawnAmountCheck

type: Number

value: 0

## MarkerSpawnList

type: List

All markers with a tag that matches the local variable *markerTag* are collected into this list at start.

## BodyList

type: List

When the character dies the body of the character is put into this list.

## MobList

type: List

If the characters spawn as a mob, when the character dies the body of the character is put into this list as well.

# Notes

* All markers must have a Box Collider with Is Trigger set to true to be able to be gathered into the SpawnMarkerList.
* Respawn must also have a Box Collider with Is Trigger set to true to be able to be gathered into the RespawnList.
* Don’t forget to set the right tag on the Respawn action.