```
local Figure = script.Parent.Parent.Parent
local Torso = Figure:WaitForChild("Torso")
local RightShoulder = Torso:WaitForChild("Right Shoulder")
local LeftShoulder = Torso:WaitForChild("Left Shoulder")
local RightHip = Torso:WaitForChild("Right Hip")
local LeftHip = Torso:WaitForChild("Left Hip")
local Neck = Torso:WaitForChild("Neck")
local Humanoid = Figure:WaitForChild("Humanoid")
local pose = "Standing"

local currentAnim = ""
local currentAnimInstance = nil
local currentAnimTrack = nil
local currentAnimKeyframeHandler = nil
local currentAnimSpeed = 1.0
local animTable = {}
local animNames = {
	idle =	{
			{ id = "http://www.roblox.com/asset/?id=180435571",
weight = 9 },
			{ id = "http://www.roblox.com/asset/?id=180435792",
weight = 1 }
		},
	walk =	{
			{ id = "http://www.roblox.com/asset/?id=180426354",
weight = 10 }
		},
	run =	{
			{ id = "run.xml", weight = 10 }
		},
	jump =	{
			{ id = "http://www.roblox.com/asset/?id=125750702",
weight = 10 }
		},
	fall =	{
			{ id = "http://www.roblox.com/asset/?id=180436148",
weight = 10 }
		},
	climb = {
			{ id = "http://www.roblox.com/asset/?id=180436334",
weight = 10 }
		},
	sit = {
			{ id = "http://www.roblox.com/asset/?id=178130996",
weight = 10 }
		},
	toolnone = {
```

```
                        { id = "http://www.roblox.com/asset/?id=182393478",
weight = 10 }
                },
        toolslash = {
                        { id = "http://www.roblox.com/asset/?id=129967390",
weight = 10 }
--                      { id = "slash.xml", weight = 10 }
                },
        toollunge = {
                        { id = "http://www.roblox.com/asset/?id=129967478",
weight = 10 }
                },
        wave = {
                        { id = "http://www.roblox.com/asset/?id=128777973", weight
= 10 }
                },
        point = {
                        { id = "http://www.roblox.com/asset/?id=128853357",
weight = 10 }
                },
        dance1 = {
                        { id = "http://www.roblox.com/asset/?id=182435998",
weight = 10 },
                        { id = "http://www.roblox.com/asset/?id=182491037",
weight = 10 },
                        { id = "http://www.roblox.com/asset/?id=182491065",
weight = 10 }
                },
        dance2 = {
                        { id = "http://www.roblox.com/asset/?id=182436842",
weight = 10 },
                        { id = "http://www.roblox.com/asset/?id=182491248",
weight = 10 },
                        { id = "http://www.roblox.com/asset/?id=182491277", weight
= 10 }
                },
        dance3 = {
                        { id = "http://www.roblox.com/asset/?id=182436935",
weight = 10 },
                        { id = "http://www.roblox.com/asset/?id=182491368",
weight = 10 },
                        { id = "http://www.roblox.com/asset/?id=182491423",
weight = 10 }
                },
        laugh = {
                        { id = "http://www.roblox.com/asset/?id=129423131", weight
= 10 }
```

```
		},
	cheer = {
			{ id = "http://www.roblox.com/asset/?id=129423030",
weight = 10 }
		},
}
local dances = {"dance1", "dance2", "dance3"}

-- Existance in this list signifies that it is an emote, the value indicates if it is a
looping emote
local emoteNames = { wave = false, point = false, dance1 = true, dance2 = true,
dance3 = true, laugh = false, cheer = false}

function configureAnimationSet(name, fileList)
	if (animTable[name] ~= nil) then
		for _, connection in pairs(animTable[name].connections) do
			connection:disconnect()
		end
	end
	animTable[name] = {}
	animTable[name].count = 0
	animTable[name].totalWeight = 0
	animTable[name].connections = {}

	-- check for config values
	local config = script:FindFirstChild(name)
	if (config ~= nil) then
--		print("Loading anims " .. name)
		table.insert(animTable[name].connections,
config.ChildAdded:connect(function(child) configureAnimationSet(name,
fileList) end))
		table.insert(animTable[name].connections,
config.ChildRemoved:connect(function(child) configureAnimationSet(name,
fileList) end))
		local idx = 1
		for _, childPart in pairs(config:GetChildren()) do
			if (childPart:IsA("Animation")) then
				table.insert(animTable[name].connections,
childPart.Changed:connect(function(property) configureAnimationSet(name,
fileList) end))
				animTable[name][idx] = {}
				animTable[name][idx]×anim = childPart
				local weightObject = childPart:FindFirstChild("Weight")
				if (weightObject == nil) then
					animTable[name][idx].weight = 1
				else
					animTable[name][idx]×weight = weightObject.Value
```

```
                        end
                        animTable[name].count = animTable[name].count + 1
                        animTable[name].totalWeight =
animTable[name].totalWeight + animTable[name][idx].weight
        --                print(name .. " [" .. idx .. "] " .. animTable[name]
[idx].anim.AnimationId .. " (" .. animTable[name][idx].weight .. ")")
                        idx = idx + 1
                end
            end
        end

        -- fallback to defaults
        if (animTable[name].count <= 0) then
            for idx, anim in pairs(fileList) do
                animTable[name][idx] = {}
                animTable[name][idx]×anim = Instance.new("Animation")
                animTable[name][idx]×anim×Name = name
                animTable[name][idx]×anim×AnimationId = anim.id
                animTable[name][idx]×weight = anim.weight
                animTable[name].count = animTable[name].count + 1
                animTable[name].totalWeight = animTable[name].totalWeight +
anim.weight
--                print(name .. " [" .. idx .. "] " .. anim.id .. " (" .. anim.weight .. ")")
            end
        end
end

-- Setup animation objects
function scriptChildModified(child)
        local fileList = animNames[child.Name]
        if (fileList ~= nil) then
            configureAnimationSet(child.Name, fileList)
        end
end

script.ChildAdded:connect(scriptChildModified)
script.ChildRemoved:connect(scriptChildModified)


for name, fileList in pairs(animNames) do
        configureAnimationSet(name, fileList)
end

-- ANIMATION

-- declarations
local toolAnim = "None"
```

```
local toolAnimTime = 0

local jumpAnimTime = 0
local jumpAnimDuration = 0.3

local toolTransitionTime = 0.1
local fallTransitionTime = 0.3
local jumpMaxLimbVelocity = 0.75

-- functions

function stopAllAnimations()
	local oldAnim = currentAnim

	-- return to idle if finishing an emote
	if (emoteNames[oldAnim] ~= nil and emoteNames[oldAnim] == false) then
		oldAnim = "idle"
	end

	currentAnim = ""
	currentAnimInstance = nil
	if (currentAnimKeyframeHandler ~= nil) then
		currentAnimKeyframeHandler:disconnect()
	end

	if (currentAnimTrack ~= nil) then
		currentAnimTrack:Stop()
		currentAnimTrack:Destroy()
		currentAnimTrack = nil
	end
	return oldAnim
end

function setAnimationSpeed(speed)
	if speed ~= currentAnimSpeed then
		currentAnimSpeed = speed
		currentAnimTrack:AdjustSpeed(currentAnimSpeed)
	end
end

function keyFrameReachedFunc(frameName)
	if (frameName == "End") then

		local repeatAnim = currentAnim
		-- return to idle if finishing an emote
		if (emoteNames[repeatAnim] ~= nil and emoteNames[repeatAnim] ==
false) then
```

```
                repeatAnim = "idle"
            end

            local animSpeed = currentAnimSpeed
            playAnimation(repeatAnim, 0.0, Humanoid)
            setAnimationSpeed(animSpeed)
        end
end

-- Preload animations
function playAnimation(animName, transitionTime, humanoid)

        local roll = math×random(1, animTable[animName].totalWeight)
        local origRoll = roll
        local idx = 1
        while (roll > animTable[animName][idx].weight) do
            roll = roll - animTable[animName][idx].weight
            idx = idx + 1
        end
--          print(animName .. " " .. idx .. " [" .. origRoll .. "]")
        local anim = animTable[animName][idx].anim

        -- switch animation
        if (anim ~= currentAnimInstance) then

            if (currentAnimTrack ~= nil) then
                currentAnimTrack:Stop(transitionTime)
                currentAnimTrack:Destroy()
            end

            currentAnimSpeed = 1.0

            -- load it to the humanoid; get AnimationTrack
            currentAnimTrack = humanoid:LoadAnimation(anim)
            currentAnimTrack.Priority = Enum.AnimationPriority.Core

            -- play the animation
            currentAnimTrack:Play(transitionTime)
            currentAnim = animName
            currentAnimInstance = anim

            -- set up keyframe name triggers
            if (currentAnimKeyframeHandler ~= nil) then
                currentAnimKeyframeHandler:disconnect()
            end
            currentAnimKeyframeHandler =
currentAnimTrack.KeyframeReached:connect(keyFrameReachedFunc)
```

```
	end

end

-------------------------------------------------------------------------------
------------
-------------------------------------------------------------------------------
------------

local toolAnimName = ""
local toolAnimTrack = nil
local toolAnimInstance = nil
local currentToolAnimKeyframeHandler = nil

function toolKeyFrameReachedFunc(frameName)
	if (frameName == "End") then
--		print("Keyframe : ".. frameName)
		playToolAnimation(toolAnimName, 0.0, Humanoid)
	end
end


function playToolAnimation(animName, transitionTime, humanoid, priority)

		local roll = math×random(1, animTable[animName].totalWeight)
		local origRoll = roll
		local idx = 1
		while (roll > animTable[animName][idx].weight) do
			roll = roll - animTable[animName][idx].weight
			idx = idx + 1
		end
--		print(animName .. " * " .. idx .. " [" .. origRoll .. "]")
		local anim = animTable[animName][idx].anim

		if (toolAnimInstance ~= anim) then

			if (toolAnimTrack ~= nil) then
				toolAnimTrack:Stop()
				toolAnimTrack:Destroy()
				transitionTime = 0
			end

			-- load it to the humanoid; get AnimationTrack
			toolAnimTrack = humanoid:LoadAnimation(anim)
			if priority then
				toolAnimTrack.Priority = priority
```

```
                end

                -- play the animation
                toolAnimTrack:Play(transitionTime)
                toolAnimName = animName
                toolAnimInstance = anim

                currentToolAnimKeyframeHandler =
toolAnimTrack.KeyframeReached:connect(toolKeyFrameReachedFunc)
        end
end

function stopToolAnimations()
        local oldAnim = toolAnimName

        if (currentToolAnimKeyframeHandler ~= nil) then
                currentToolAnimKeyframeHandler:disconnect()
        end

        toolAnimName = ""
        toolAnimInstance = nil
        if (toolAnimTrack ~= nil) then
                toolAnimTrack:Stop()
                toolAnimTrack:Destroy()
                toolAnimTrack = nil
        end


        return oldAnim
end

-------------------------------------------------------------------------------
------------
-------------------------------------------------------------------------------
------------


function onRunning(speed)
        if speed > 0.01 then
                playAnimation("walk", 0.1, Humanoid)
                if currentAnimInstance and currentAnimInstance×AnimationId ==
"http://www.roblox.com/asset/?id=180426354" then
                        setAnimationSpeed(speed / 14.5)
                end
                pose = "Running"
        else
                if emoteNames[currentAnim] == nil then
```

```
                playAnimation("idle", 0.1, Humanoid)
                pose = "Standing"
            end
        end
end

function onDied()
        pose = "Dead"
end

function onJumping()
        playAnimation("jump", 0.1, Humanoid)
        jumpAnimTime = jumpAnimDuration
        pose = "Jumping"
end

function onClimbing(speed)
        playAnimation("climb", 0.1, Humanoid)
        setAnimationSpeed(speed / 12.0)
        pose = "Climbing"
end

function onGettingUp()
        pose = "GettingUp"
end

function onFreeFall()
        if (jumpAnimTime <= 0) then
                playAnimation("fall", fallTransitionTime, Humanoid)
        end
        pose = "FreeFall"
end

function onFallingDown()
        pose = "FallingDown"
end

function onSeated()
        pose = "Seated"
end

function onPlatformStanding()
        pose = "PlatformStanding"
end

function onSwimming(speed)
        if speed > 0 then
```

```
            pose = "Running"
        else
            pose = "Standing"
        end
end

function getTool()
        for _, kid in ipairs(Figure:GetChildren()) do
            if kid.className == "Tool" then return kid end
        end
        return nil
end

function getToolAnim(tool)
        for _, c in ipairs(tool:GetChildren()) do
            if c.Name == "toolanim" and c.className == "StringValue" then
                return c
            end
        end
        return nil
end

function animateTool()

        if (toolAnim == "None") then
            playToolAnimation("toolnone", toolTransitionTime, Humanoid,
Enum.AnimationPriority.Idle)
            return
        end

        if (toolAnim == "Slash") then
            playToolAnimation("toolslash", 0, Humanoid,
Enum.AnimationPriority.Action)
            return
        end

        if (toolAnim == "Lunge") then
            playToolAnimation("toollunge", 0, Humanoid,
Enum.AnimationPriority.Action)
            return
        end
end

function moveSit()
        RightShoulder.MaxVelocity = 0.15
        LeftShoulder.MaxVelocity = 0.15
        RightShoulder:SetDesiredAngle(3.14 /2)
```

```
	LeftShoulder:SetDesiredAngle(-3.14 /2)
	RightHip:SetDesiredAngle(3.14 /2)
	LeftHip:SetDesiredAngle(-3.14 /2)
end

local lastTick = 0

function move(time)
	local amplitude = 1
	local frequency = 1
	local deltaTime = time - lastTick
	lastTick = time

	local climbFudge = 0
	local setAngles = false

	if (jumpAnimTime > 0) then
		jumpAnimTime = jumpAnimTime - deltaTime
	end

	if (pose == "FreeFall" and jumpAnimTime <= 0) then
		playAnimation("fall", fallTransitionTime, Humanoid)
	elseif (pose == "Seated") then
		playAnimation("sit", 0.5, Humanoid)
		return
	elseif (pose == "Running") then
		playAnimation("walk", 0.1, Humanoid)
	elseif (pose == "Dead" or pose == "GettingUp" or pose == "FallingDown"
or pose == "Seated" or pose == "PlatformStanding") then
--		print("Wha " .. pose)
		stopAllAnimations()
		amplitude = 0.1
		frequency = 1
		setAngles = true
	end

	if (setAngles) then
		local desiredAngle = amplitude × math×sin(time × frequency)

		RightShoulder:SetDesiredAngle(desiredAngle + climbFudge)
		LeftShoulder:SetDesiredAngle(desiredAngle - climbFudge)
		RightHip:SetDesiredAngle(-desiredAngle)
		LeftHip:SetDesiredAngle(-desiredAngle)
	end

	-- Tool Animation handling
	local tool = getTool()
```

```
	if tool and tool:FindFirstChild("Handle") then

		local animStringValueObject = getToolAnim(tool)

		if animStringValueObject then
			toolAnim = animStringValueObject×Value
			-- message recieved, delete StringValue
			animStringValueObject×Parent = nil
			toolAnimTime = time + .3
		end

		if time > toolAnimTime then
			toolAnimTime = 0
			toolAnim = "None"
		end

		animateTool()
	else
		stopToolAnimations()
		toolAnim = "None"
		toolAnimInstance = nil
		toolAnimTime = 0
	end
end

-- connect events
Humanoid.Died:connect(onDied)
Humanoid.Running:connect(onRunning)
Humanoid.Jumping:connect(onJumping)
Humanoid.Climbing:connect(onClimbing)
Humanoid.GettingUp:connect(onGettingUp)
Humanoid.FreeFalling:connect(onFreeFall)
Humanoid.FallingDown:connect(onFallingDown)
Humanoid.Seated:connect(onSeated)
Humanoid.PlatformStanding:connect(onPlatformStanding)
Humanoid.Swimming:connect(onSwimming)

-- setup emote chat hook
--[[game:GetService("Players").LocalPlayer.Chatted:connect(function(msg)
	local emote = ""
	if msg == "/e dance" then
		emote = dances[math.random(1, #dances)]
	elseif (string×sub(msg, 1, 3) == "/e ") then
		emote = string.sub(msg, 4)
	elseif (string×sub(msg, 1, 7) == "/emote ") then
		emote = string.sub(msg, 8)
	end
```

```
	if (pose == "Standing" and emoteNames[emote] ~= nil) then
		playAnimation(emote, 0.1, Humanoid)
	end

end)--]]


-- main program

-- initialize to idle
playAnimation("idle", 0.1, Humanoid)
pose = "Standing"

while Figure.Parent ~= nil do
	local _, time = wait(0.1)
	move(time)
end



------Respawn------
local marine = script.Parent.Parent.Parent
local myHead = marine.Head
local myHuman = marine.Humanoid

local modules = script.Parent.Parent.Modules
local core = require(modules×Core)
local actions = require(modules×Actions)
local combat = require(modules×Combat)


--Sound variables
local runningSound = myHead.Running
local jumpingSound = myHead.Jumping

--myHuman:SetStateEnabled(Enum.HumanoidStateType.Flying,false)
--myHuman:SetStateEnabled(Enum.HumanoidStateType.Physics,false)
--myHuman:SetStateEnabled(Enum.HumanoidStateType.Ragdoll,false)
--
myHuman:SetStateEnabled(Enum.HumanoidStateType.RunningNoPhysics,false
)
--
myHuman:SetStateEnabled(Enum.HumanoidStateType.StrafingNoPhysics,false)
--myHuman:SetStateEnabled(Enum.HumanoidStateType.Swimming,false)

myHuman.Running:Connect(function(speed)
```

```
	if speed>0 then
		runningSound:Play()
	else
		runningSound:Stop()
	end
end)

myHuman.Jumping:Connect(function()
	jumpingSound:Play()
end)

myHuman.PlatformStanding:Connect(function(platStand)
	if platStand then
		runningSound:Stop()
	end
end) --16157972 <- Trash removed.

local hair =
{32278814,13477818,80922374,376548738,16630147,15913837,5064651922,
83013207,12270248,26658141,
	188003563,13655562,4735347390,4735346175}
local facial =
{5355727732,5355564336,4510537113,5164598367,4528880486,45452945
88,11884330,20642008,74970669,

4507911797,5231192146,158066137,5030224026,5031008665,4102114619,49
95497755,987022351,5414672551,4875925250,

4238241252,4904398878,5231324178,4940496302,5167393248,493354964
9,4786863817,5063784744,4708784614,74221074,
	5166827175,4524490255,4875895114,5197016342}

local marine = script.Parent.Parent.Parent

function applyAsset(id)
	local item = game:GetService("InsertService"):LoadAsset(id)
	local items = item:GetChildren()
	local accessory = items[1]
	local handle = accessory×Handle

	accessory×Parent = marine
	item:Destroy()

	task.wait()

	handle×Parent = marine
	handle×Name = id
```

```
		accessory:Destroy()

		local weld = Instance×new("WeldConstraint")
		weld×Part0 = marine.Head
		weld×Part1 = handle
		weld×Parent = handle
end

if script.Parent.Parent.Parent.Settings.RandomAppearance.Value then

		applyAsset(facial[math.random(#facial)])

		if math×random(4) == 1 then
			marine.ACH:Destroy()
			applyAsset(hair[math.random(#hair)])
		end

		local skinOptions =
{"Wheat","Cashmere","Beige","Khaki","Buttermilk","Pastel yellow"}
		local randomSkin =
BrickColor.new(skinOptions[math.random(#skinOptions)])
		local bc = marine["Body Colors"]
		bc×HeadColor = randomSkin
		bc×LeftArmColor = randomSkin
		bc×RightArmColor = randomSkin
		bc×LeftLegColor = randomSkin
		bc×RightLegColor = randomSkin
		bc×TorsoColor = randomSkin
end

local marine = script.Parent.Parent.Parent
local clone = marine:Clone()

------Fix network issues------
pcall(function()
		while marine.HumanoidRootPart:GetNetworkOwnershipAuto() do
			marine.HumanoidRootPart:SetNetworkOwner(nil)
			task.wait()
		end
end)

-- Tag NPC with it's team so it can be referenced and communicated with by
other NPCs.
local CollectionService = game:GetService("CollectionService")
CollectionService:AddTag(marine, marine.Settings.Team.Value)
```

```
-----Settings------
local mySettings = marine.Settings

local meleeRange = mySettings.MeleeRange.Value -- range at which the marine
will do melee
local ignoreFolder = mySettings.IgnoreFolder.Value
if ignoreFolder == nil then
	if workspace:FindFirstChild("MarineIgnoreFolder") then
		ignoreFolder = workspace.MarineIgnoreFolder
	else
		local newIgnoreFolder = Instance×new("Folder",workspace)
		newIgnoreFolder×Name = "MarineIgnoreFolder"
		ignoreFolder = newIgnoreFolder
	end
end
mySettings×IgnoreFolder×Value = ignoreFolder

------Modules------
local modules = script.Parent.Parent.Modules
local core = require(modules×Core)
local actions = require(modules.Actions)
local combat = require(modules.Combat)
local targeting = require(modules.Target)
local movement = require(modules.Movement)
local debugger = require(modules.Debug)
local vehicle = require(modules×Vehicle)
local gunner = require(modules.Gunner)
local pathing = require(modules×Pathing)
local NPC = require(modules×NPC)
local Passenger = require(modules.Passenger)

------Body Variables------
local myHuman = marine.Humanoid
local myRoot = marine.HumanoidRootPart
local myHead = marine.Head

-- What the heck does this do...
local rotAttach1 = Instance.new("Attachment")
rotAttach1.Parent = workspace.Terrain

-- Sounds
local hurtSound = myHead.Hurt
local diedSound = myHead.Died

-----OOP Stuff------
```

```
local npc = NPC×new{
	Instance = script.Parent.Parent.Parent
}

npc:Inherit(actions)
npc:Inherit(movement)
npc:Inherit(targeting)
npc:Inherit(combat)

local function vehicleLogic(targetDist)
	if npc×Vehicle×Role == "Driver" then


		-- Don't start driving to the target until we have all of our passengers.
		if npc.Vehicle:PickupNearestAlly() then

			task.wait(1)

		elseif targetDist > npc.Settings.Vehicle.MinDrivingDist.Value then
			-- If we do have a vehicle determine how to drive there or to exit
etc
			if npc.Vehicle:DriveTo(npc:GetTarget().Position,npc:GetTarget())
then

			elseif not npc.Vehicle.Instance:GetAttribute("Gunner") then
				npc.Vehicle:Blacklist(npc:GetTarget())
				npc×Vehicle = npc.Vehicle:Exit()

			else
				npc.Vehicle:GetFree()
			end

		elseif not npc.Vehicle.Instance:GetAttribute("Gunner") and
npc.Vehicle:GetSpeed() < 2 then
			-- Exit vehicle
			-- TODO: Never exits when gunner is present. Eventually they
should probably get out.
			npc×Vehicle = npc.Vehicle:Exit()

		elseif npc.Vehicle.Instance:GetAttribute("Gunner") and targetDist <
npc.Settings.Vehicle.MinDrivingDist.Value / 1.5 then
			-- This case is when we have a gunner and we are too close to
the enemy.


			-- Is the enemy more towards the front of us or the back?
```

```
                local enemyYaw =
math.abs(core.findYaw(npc.Vehicle.Body.Position, npc:GetTarget().Position,
npc.Vehicle.Body))

                -- Maybe we can just reverse straight back from the enemy?
                if not npc.Vehicle:CheckBumper(true) and enemyYaw < 90 then
                    npc.Vehicle:TurnTowardsAngle(0)
                    npc.Vehicle:SmartGo(-0.4, 1)


                    -- If the back bumper is clear of obstacles we can back
away.
                elseif not npc.Vehicle:CheckBumper() and enemyYaw > 90 then
                    npc.Vehicle:TurnTowardsAngle(0)
                    npc.Vehicle:SmartGo(1, 1)

                else
                    -- TODO: Need some way of finding a location to drive to
                    task.wait(1)
                end
            end
        elseif npc×Vehicle×Role == "Gunner" then
            -- TODO: Determine when we should exit the gun
            -- Exit when there is no target sight for x amount of time and there is
no driver.
            if not npc.Vehicle.Instance:GetAttribute("Driver") and tick() -
npc:GetTargetLastSeen() > npc.Settings.Vehicle.Patience.Value and
npc.Vehicle:GetSpeed() < 2  then
                npc×Vehicle = npc.Vehicle:Exit()
            elseif npc:GetTarget() and core.checkDist(npc:GetTarget(),npc.Root)
< 7 and npc.Vehicle:GetSpeed() < 2 then
                npc×Vehicle = npc.Vehicle:Exit()
            else
                task.wait(0.5) -- 2
            end
        elseif npc×Vehicle×Role == "Passenger" then
            if npc:CheckSightToTarget() and npc.Vehicle:GetSpeed() < 2 and
npc:CanExitVehicle() then
                npc×Vehicle = npc.Vehicle:Exit()
            end
        else
            task.wait(1)
        end
end

local function getVehicleSpot(targetDist)
        -- Try to get in a gun position if there is a Warthog with a driver.
```

```
	local veh, vehDist = vehicle.findNearest(npc, true, "Gunner")
	if veh and vehDist <= targetDist then

		npc×Vehicle = gunner.new(veh, npc)

		local bumper = veh.Body.Body.RearBumper

		npc:PathToLocation(bumper)

		if ((npc.Vehicle.Body.Position - npc.Root.Position).Magnitude <=
npc.Settings.Vehicle.EnterDist.Value or
			(bumper.WorldPosition - npc.Root.Position).Magnitude <=
npc.Settings.Vehicle.EnterDist.Value) then
			npc.Vehicle:Enter()
		else
			npc×Vehicle = npc.Vehicle:Destroy()
		end
		return
	end

	local veh, vehDist = vehicle.findNearest(npc, true, "Passenger")
	if veh and vehDist <= targetDist then

		npc×Vehicle = Passenger.new(veh, npc)

		local door = veh.Body.Body.SideDoor

		npc:PathToLocation(door)

		if ((npc.Vehicle.Body.Position - npc.Root.Position).Magnitude <=
npc.Settings.Vehicle.EnterDist.Value or
			(door.WorldPosition - npc.Root.Position).Magnitude <=
npc.Settings.Vehicle.EnterDist.Value) then
			npc.Vehicle:Enter()
		else
			npc×Vehicle = npc.Vehicle:Destroy()
		end
		return
	end

	-- We don't have a vehicle do we need one?
	local veh, vehDist = vehicle.findNearest(npc, false, "Driver")

	-- Try to generate a path with it
	-- Kind of a waste that we don't use this path.

	if veh and vehDist <= targetDist then
```

```
            npc×Vehicle = vehicle.new(veh, npc)

            local vehPath = pathing.getVehiclePath(veh, npc:GetTarget().Position,
20)

            if vehPath then

                  local door = veh.Body.Body.FrontDoor

                  if (npc.Vehicle.Body.Position - npc.Root.Position).Magnitude >=
npc.Settings.Vehicle.EnterDist.Value then
                        -- Walk to the vehicle

                        -- Set that the driver position is now occupied so that other
NPCs don't try to steal it.
                        if not npc:PathToLocation(door) then
                              -- No valid path to the vehicle, blacklist it.
                              npc.Vehicle:Blacklist(npc:GetTarget())
                        end
                  end

                  -- Sit when we are close enough
                  if (door.WorldPosition - npc.Root.Position).Magnitude <=
npc.Settings.Vehicle.EnterDist.Value
                        and not npc:CheckSightToTarget() then
                        npc.Vehicle:Enter()
                  else
                        npc×Vehicle = npc.Vehicle:Destroy()
                  end

            else
                  npc×Vehicle = npc.Vehicle:Destroy()
                  npc:PathToLocation(npc:GetTarget())
            end
      else
            npc:PathToLocation(npc:GetTarget())
      end
end


local idleIndex = 5
local function movementLogic(target)
      local targetDist = core.checkDist(target,npc.Root)

      local canSee = npc:CheckSightToTarget()
```

```
	idleIndex -= 1
	if canSee then
		npc:SetTargetLastSeen()
		idleIndex = 5 --100
	end

	-- Vehicle logic first
	if npc.Vehicle and npc.Vehicle:IsInVehicle() and npc.Vehicle:Validate() then
		vehicleLogic(targetDist)

	elseif (targetDist > 50 or not canSee) and (myRoot.Position.Y <
target.Position.Y-10 or idleIndex <= 0) then
		-- This case we need to either drive to the target or pathfind to it.

		if core.runAtTS("SoldierAICheckForVehicle", 5) and targetDist >
npc.Settings.Vehicle.MinDrivingDist.Value and npc:CanEnterVehicle() then
			getVehicleSpot(targetDist)
		else
			npc:PathToLocation(target)
		end

	elseif (targetDist > 30 or not canSee) and core.checkPotentialSight(target)
then
		npc:SlowAdvance()
		task.wait(0.3)

	elseif npc.Settings.Movement.CanCrouch.Value and targetDist > 50 and
canSee and npc:GetM4Equipped() and math×random(5) == 1
		and core.isLine(npc.Root.Position - Vector3.new(0,1.5,0),
target.Position, {npc.Instance, target.Parent}) then
		npc:Crouch()
		-- Busy waiting... sue me
		local startedCrouch = core×tick()
		while npc:CheckSightToTarget() and core.checkDist(npc.Root,target)
> 50 do
			if core.tick() - startedCrouch > 5 and math×random(3) == 1 then
				break
			end
			task.wait(0.5)
		end
		npc:Stand()

	elseif npc.Settings.Movement.CanStrafe.Value and targetDist > 50 and
targetDist < 90 and canSee and npc:GetM4Equipped() then
		npc:Strafe()
		task.wait(0.5)
```

```
	elseif npc.Settings.Movement.CanBackpedal.Value and targetDist > 4 and
targetDist < 50 and canSee
		and math.abs(myRoot.Position.Y - target.Position.Y) < 2 and
(npc:GetM4Aimed() or npc:GetReloading()) then
		npc:Retreat()
		task.wait(0.2)

	elseif (targetDist < 4 and canSee) or npc:GetKnifeEquipped() then
		npc:ChaseTargetDirectly()
	else
		task.wait(0.5)
	end
end


local function movementLoop()
	while myHuman.Health>0 do
		local target = npc:GetTarget()
		if target then
			movementLogic(target)
		else
			if mySettings.Wander.Value and math×random(4) == 3 then
				npc:WalkRandom()
			end
			task.wait(3)
		end
		task.wait(0.1)
	end
end

local function searchTargetLoop()
	while myHuman.Health>0 do

		-- Updates the target based on any that are really close to the NPC.
		-- Looks good for quick updates when melee units are attacking.
		if core.runAtTS("GetIncomingAttackers",2) then
			npc:GetIncomingAttackers()
		end

		-- Keep track of what vehicle the target is potentially using.
		-- We need to know so we can ignore it during raycasting.
		if npc:GetTarget() and core.runAtTS("UpdateEnemyVehicle",2) then

npc:SetTargetVehicle(core.getEnemyVehicleInstance(npc:GetTarget()))
		end

		-- Find a new target if the current one is not valid
```

```
		-- Also find a new target if we are hurt. This should make the marine
look more reactionary to damage.
		if not npc:CheckTarget() or npc:GetTookDamage() then
			npc:SetTookDamage(false)
			--npc:SetTarget(nil)
			npc:FindTarget()

		-- Find a new target if we can't see the target try finding one we can
see.
		elseif not npc:CheckSightToTarget() and npc:GetSeeTarget() then
			npc:FindTarget()
			if not npc:GetTarget() then
				task.wait(3)
			end
		end
		task.wait(0.5)
	end
end

local function aimingLoop()
	while myHuman.Health>0 do

		-- Vehicle integrity check
		if npc.Vehicle and npc.Vehicle:IsInVehicle() then
			if not npc.Vehicle:Validate() then
				npc×Vehicle = nil
			end
		end

		if npc:GetTarget() then
			if npc:CheckSightToTarget() and npc:GetM4Equipped() then
				npc:Aim()
			else
				task.wait(0.5)
			end
		else
			task.wait(2)
		end
		task.wait(0.1)
	end
end

local function attackLogic()
	npc:UpdateFace("Attacking")

	local distance = core.checkDist(myRoot,npc:GetTarget())
	if distance > meleeRange then
```

```
            -- If there is a cluster of enemies far enough away throw grenade.
            -- 5/20/22 added randomness, probably gotta tweak
            if npc:CheckClusterAtTarget() and distance < 100 and distance > 40
and npc:GetGrenadeCool() and math×random(10) == 1  then
                  npc:ThrowGrenade(npc:GetTarget())
            elseif npc:GetM4Equipped() then
                  if npc:FacingTarget() and npc:CheckShot() then
                        npc:Shoot(npc:GetTarget())
                  end
            else
                  npc:DrawM4()
            end
      else
            npc:DrawKnife()
            npc:Stab()
      end
end

local function turretLogic()
      if npc:CheckTarget() and npc.Vehicle:CheckSightTurret() then
            npc.Vehicle:StartAimingTurret()
            while npc.Vehicle.Aiming and not npc.Vehicle.Shooting do
                  task.wait(0.1)
                  if npc.Vehicle:CheckAim() and npc.Vehicle:IsTargetInRange()
then
                        npc.Vehicle:StartShootingTurret()
                  end
            end

            if not
npc:CheckSightToTarget(npc.Settings.Vehicle.MaxTurretRange.Value) then
                  -- Once we start shooting don't stop until we can't see it.
                  -- I don't care about hitting every shot it's a machine gun.
                  npc.Vehicle:StopShootingTurret()
            end
      else
            npc.Vehicle:StopShootingTurret()
      end
      task.wait(1)
end

local function attackLoop()
      while myHuman.Health>0 do

            if not npc.Vehicle or not npc.Vehicle:IsInVehicle() then
```

```
                if npc:GetTarget() then
                        npc:UpdateFace("Hunting")

                        -- If we have a target we can see
                        if npc:CheckSightToTarget() and npc:CheckTarget() and
core.checkDist(myRoot,npc:GetTarget()) < mySettings.M4.Range.Value then
                                attackLogic()
                        elseif npc:GetM4Equipped() and not npc:GetM4Aimed()
then
                                npc:LowerM4()
                        end
                else
                        task.wait(2)
                        npc:UpdateFace("Idle")
                        npc:YieldWeapons()
                end
        elseif npc×Vehicle×Role == "Gunner" then
                turretLogic()
        else
                task.wait(1)
        end
        task.wait(0.3)
    end
end

task.spawn(searchTargetLoop)
task.spawn(attackLoop)
task.spawn(movementLoop)
task.spawn(aimingLoop)

-- Soldier COMMs
-- Found out you can't pass tables with metamethods using this.
marine.Query.OnInvoke = function()
    local cleanNPC = {}
    if npc.Vehicle then
        cleanNPC×Vehicle = {}
        cleanNPC×Vehicle×Instance = npc.Vehicle.Instance
        cleanNPC×Vehicle×Role = npc.Vehicle.Role
        cleanNPC×Vehicle×InVehicle = npc.Vehicle:IsInVehicle()
    else
        cleanNPC×Vehicle = nil
    end

    cleanNPC×Root = npc.Root

    return cleanNPC
end
```

```
myHuman.Died:Connect(function()
	if npc.Vehicle then
		if npc.Vehicle:IsInVehicle() then
			npc.Vehicle:Exit()
		else
			npc.Vehicle:Destroy()
		end
	end
	npc×Torso×AssemblyLinearVelocity  =
Vector3.new(math.random(-25,25),math.random(50,75),math.random(-25,25))
	diedSound:Play()
	npc:YieldM4()
	npc:UpdateFace("Dead")
	for i,v in ipairs(marine:GetDescendants()) do
		if v:IsA("BallSocketConstraint") then
			v.Enabled = true
		elseif v:IsA("BasePart") and v.Name ~= "myHumanoidRootPart" then
			v.CanCollide = false
		elseif v:IsA("Motor6D") then
			v:Destroy()
		end
	end
	npc×Head×CanCollide = true
	if marine.Settings.Respawn.Value then
		task.wait(marine.Settings.RespawnDelay.Value)
		clone.Parent = marine.Parent
	end
	for i,v in ipairs(marine:GetDescendants()) do
		if v:IsA("BasePart") or v:IsA("Decal") then
			game:GetService("TweenService"):Create(v,TweenInfo.new(0.2),
{Transparency = 1}):Play()
		end
	end
	rotAttach1:Destroy()
	task.wait(0.2)
	marine:Destroy()
end)

local oldHealth = myHuman.Health
local soundSpeeds = {0.9,0.95,1,1.05,1.1}
myHuman.HealthChanged:Connect(function(health)
	if health < oldHealth and hurtSound.IsPlaying == false then
		npc:SetTookDamage(true)
		if math×random(3) == 1 then
			hurtSound.PlaybackSpeed =
soundSpeeds[math.random(#soundSpeeds)]
```

```
                hurtSound:Play()
            end
            task.spawn(function()
                npc:UpdateFace("Hurt")
                task.wait(1)
                if myHead:FindFirstChild("faceHurt") then
                    npc:UpdateFace(npc:GetMood(),true)
                end
            end)
        end
        oldHealth = health
end)
```