```
local Vehicle = {}
Vehicle.__index = Vehicle

local CollectionService = game:GetService("CollectionService")
local RunService = game:GetService("RunService")

-- Constants
MARINE_VEHICLE_TAG = "MarineVehicle"

-- Modules
local pathing = require(script×Parent×Pathing)
local debugger = require(script.Parent.Debug)
local core = require(script×Parent×Core)


function Vehicle.new(vehicle, npc)

	local o = {}
	o.Instance = vehicle
	o.Occupant = npc
	o.Body = vehicle.Body.Body
	o.FrontBumper = vehicle.Body.Body.FrontBumper
	o.RearBumper = vehicle.Body.Body.RearBumper
	o.Seat = vehicle.VehicleSeat
	o.Team = vehicle.Settings.Team
	o.InVehicle = false
	o.Role = "Driver"

	vehicle:SetAttribute(o.Role, true)

	setmetatable(o, Vehicle)

	o:SetTeam(npc.Team)

	return o
end

function Vehicle:Destroy()
	self.Instance:SetAttribute(self.Role, false)
	if not self.Instance:GetAttribute("Driver") and not
self.Instance:GetAttribute("Gunner") and not
self.Instance:GetAttribute("Passenger") then
		self:SetTeam("")
	end
	self = nil
end
```

```
function Vehicle:Enter()

	self.Occupant:YieldWeapons()
	self.Seat:Sit(self.Occupant.Human)

	RunService.Heartbeat:Wait()
	RunService.Heartbeat:Wait()

	self×Occupant×LeftShoulder.Enabled = false
	self×Occupant×RightShoulder.Enabled = false
	self×Occupant×ShoulderHingeR.Enabled = true
	self×Occupant×ShoulderHingeL.Enabled = true

	self.InVehicle = true
end


function Vehicle:Exit()
	self.Instance:SetAttribute("Failures",0)

	self×Occupant×ShoulderHingeR.Enabled = false
	self×Occupant×ShoulderHingeL.Enabled = false
	self×Occupant×LeftShoulder.Enabled = true
	self×Occupant×RightShoulder.Enabled = true

	-- TODO: Why is this set here but not in Enter?
	if self.Seat.Parent then
		pcall(function()
			self.Seat:SetNetworkOwnershipAuto()
		end)
	end
	self.Occupant:SetVehicleTimestamp()
	self.Occupant.Human.Sit = false

	self:Destroy()

	return nil
end

function Vehicle:IsInVehicle()
	return self.InVehicle
end

function Vehicle:Go(speed: number)
	self.Seat.ThrottleFloat = speed
end
```

```
function Vehicle:SmartGo(speed, length)

	local backwards = false
	if speed < 0 then
		backwards = true
	end

	self:Go(speed)

	for i=1, length*10 do
		if self:CheckBumper(backwards) then
			break
		end
		task.wait(0.1)
	end

	self:Go(0)
	self:TurnTowardsAngle(0)
end

-- pos is the position of the NPC root
-- Returns a tuple of the Vehicle and distance
function Vehicle.findNearest(npc: ObjectValue, requireDriver: BoolValue, role:
string)
	local Vehicles = CollectionService:GetTagged(MARINE_VEHICLE_TAG)
	local bestDist = math.huge
	local bestVeh = nil
	for i, tempVehicle in ipairs(Vehicles) do

		if Vehicle.isValid(npc, tempVehicle) and
tempVehicle:GetAttribute(role) == false
			and not Vehicle.isBlacklisted(npc, tempVehicle) and (not
requireDriver or tempVehicle:GetAttribute("Driver")) then

			local tempDist = (npc×Root×Position -
tempVehicle×Body×Body×Position)×Magnitude
			if tempDist < bestDist then
				bestDist = tempDist
				bestVeh = tempVehicle
			end
		end
	end

	return bestVeh, bestDist
end
```

```
-- Includes NPC related checks once using the vehicle.
-- Will automatically correct weird stuff that happens while driving the vehicle.
-- TODO: Add checks that include the vehicle flipping, blown up, missing
wheels
function Vehicle:Validate()
	if self.Seat.Occupant ~= self.Occupant.Human or not
Vehicle.isValid(self.Occupant, self.Instance) then
		self:Exit()
		return false
	end
	return true
end

function Vehicle:GetFree()
	if not self:CheckBumper(true) then
		-- If we can reverse then try it.
		self:TurnRandom()
		self:SmartGo(-0.4,2)

	elseif not self:CheckBumper() then
		-- If we can move forward then try it.
		self:TurnRandom()
		self:SmartGo(0.4,2)
	end
end

function Vehicle.isFree(veh)
	return not veh:GetAttribute("Driver")
end

function Vehicle.isValid(npc,veh)
	-- TODO: More logic like checking the tire status, if it is stuck,
upsidedown, destroyed etc
	local bodyModel = veh:FindFirstChild("Body")
	return veh.Parent and bodyModel and bodyModel:FindFirstChild("Body")
and veh.Humanoid.Health > 0
		and bodyModel.Body.CFrame.UpVector.Y > 0.5 and
Vehicle.checkTeam(npc, veh)
end

-- Returns true if blocked
function Vehicle:CheckBumper(back)

	local rayParams = RaycastParams×new()
	rayParams×FilterDescendantsInstances =
{self.Occupant.Instance,self.Instance}
```

```
	local VehicleLength = self.Instance:GetExtentsSize().Z
	local VehicleWidth = math×ceil(self×Instance:GetExtentsSize().X/3)
	for i = -VehicleWidth, VehicleWidth do

		--Raising bumper slightly to avoid weirdness on uneven ground
		local origin = (self×FrontBumper×WorldCFrame ×
CFrame×new(i,1,3))×Position
		local dir = self×FrontBumper×WorldCFrame×LookVector×3
		if back then
			origin = (self×RearBumper×WorldCFrame ×
CFrame×new(i,1,3))×Position
			dir = -dir
		end



		local result = workspace:Raycast(origin,dir,rayParams)
		debugger.showRay(origin,dir)
		if result and result.Instance then
			--return result.Instance
			return true
		end

		-- Check downwards to ensure we don't drive off a cliff (hopefully)
		origin += dir
		dir = Vector3.new(0,-5,0)
		result = workspace:Raycast(origin,dir,rayParams)
		debugger.showRay(origin,dir)
		if not result then
			return true
		end
	end
end

function checkBrokenPoints(veh, path)

	local waypoints = path:GetWaypoints()
	local confirmedWaypoints = {}

	for i,waypoint in ipairs(waypoints) do

		--Raycast below each waypoint to ensure there is ground underneath
		--Prevents waypoints that are glitched under the map
		--5/7/21 Fixed bug on baseplate maps. Waypoints were slightly too
low causing the whole path
		--to return as broken. Fixed this by raising the initial waypoint height
by half a stud.
		local rayParams = RaycastParams×new()
```

```
            rayParams×FilterDescendantsInstances = {veh.Occupant.Instance}
            local result =
workspace:Raycast(waypoint.Position+Vector3.new(0,0.5,0),Vector3.new(0,-5,0
), rayParams)
            if result then
                  table.insert(confirmedWaypoints,waypoint)
            end
      end

      return confirmedWaypoints
end

function findTurns(veh, path)

      --Found line from waypoint 1 to 40
      --Waypoints are in a line if x or z value changes very little over a period of
time

      local waypoints = checkBrokenPoints(veh, path)
      local currentDifference = Vector2.new(0,0)

      local importantWaypoints = {}

      --First find which axis the line is by finding the axis with the smallest
difference
      --Then loop through next points and check axis until axis has a difference
> 1
      for i,waypoint in ipairs(waypoints) do
            if i ~= #waypoints then
                  local difference =
(Vector2.new(waypoint×Position×X,waypoint.Position.Z) -
Vector2.new(waypoints[i+1].Position.X,waypoints[i+1].Position.Z))
                  if ((currentDifference - difference).Magnitude) > 0.5 then
                        debugger.mark(waypoint.Position)
                        table.insert(importantWaypoints,waypoint)
                  end
                  currentDifference = difference
            end
      end

      --Ensure final point is included also
      table.insert(importantWaypoints,waypoints[#waypoints])

      return importantWaypoints
end

function Vehicle:IsAtWaypoint(pos)
```

```
	return (pos - self.FrontBumper.WorldPosition).Magnitude <
self.Occupant.Settings.Vehicle.MaxWaypointDist.Value
end

function Vehicle:TurnTowardsAngle(desiredAngle)

	self.Occupant.ShoulderHingeR.TargetAngle = -desiredAngle
	self.Occupant.ShoulderHingeL.TargetAngle = desiredAngle

	--Changed 15 to 30 turning too sharply before
	local turndir = desiredAngle/55

	self.Seat.SteerFloat = turndir

	return desiredAngle
end

local maxSpeed = 0.5
function Vehicle:DriveTowards(pos)

	local currentDist = (self.FrontBumper.WorldPosition - pos).Magnitude

	local desiredAngle =
core×findYaw(self×FrontBumper×WorldPosition,pos,self.Occupant.Root)


	--Changed from 20 to 100 to make faster turns
	local speed = math×abs(50/desiredAngle)

	-- Reverse if too tight of a turn
	if (desiredAngle > 100 or desiredAngle < -100) and not
self:CheckBumper(true) then

		self:SmartReverse(pos,-desiredAngle)
	end

	self:TurnTowardsAngle(desiredAngle)

	--self.Seat.ThrottleFloat = speed
	self:Go(speed)
end

-- The goal of this is to reverse while our angle is getting smaller.
-- Angle of 0 should be the front so we should continue until it doesn't get
closer to this value.
function Vehicle:SmartReverse(pos, turnAngle, minReverseTime)
```

```
	minReverseTime = minReverseTime or 0

	--self.Seat.ThrottleFloat = -0.4
	self:Go(-0.4)

	-- Should allow the NPC to back straight up into a waypoint
	if math.abs(turnAngle) > 170 then
		turnAngle = 0
	end



	self:TurnTowardsAngle(turnAngle)

	local currentAngle =
math×abs(core×findYaw(self×FrontBumper×WorldPosition,pos,self.Occupant.R
oot))
	local prevAngle = math.huge

	-- Continously backsup until we are being counter productive or at the
waypoint.
	local lastMove = tick()
	while currentAngle <= prevAngle and not self:IsAtWaypoint(pos) do
		prevAngle = currentAngle
		currentAngle =
math.abs(math.abs(core.findYaw(self.FrontBumper.WorldPosition,pos,self.Occu
pant.Root)))

		if self:GetSpeed() > 2 then
			lastMove = tick()
		end

		if currentAngle < 20 then
			break
		elseif tick() - lastMove > 0.5 then
			self:Go(1)
			task.wait(0.5)
			break
		elseif self:CheckBumper(true) then
			break
		elseif not self:Validate() then
			break
		end
		task.wait(0.2)
	end

	-- Ensure we backup for the minimum amount time
	task.wait((tick() + minReverseTime) - tick())
```

```
end

function Vehicle:TurnRandom()
	local turn = 45
	if math×random(2) == 1 then
		turn = -turn
	end
	self:TurnTowardsAngle(turn)
end

function Vehicle:Reverse(length)


	--self.Seat.ThrottleFloat = -0.2
	self:Go(-0.4)

	for i=1, length*10 do
		if self:CheckBumper(true) then
			break
		end
		task.wait(0.1)
	end

	self:Go(0)
	self:TurnTowardsAngle(0)
end

local pathFailures = 0
local maxPathFailures = 3
function Vehicle:DrivePath(path,instance)

	-- Now we can determine seperate logic if we are driving to an instance
such as
	-- our ally or our enemy.
	local currentTarget = nil
	local toTarget = false
	if instance == self.Occupant:GetTarget() then
		currentTarget = instance
		toTarget = true
	end

	local waypointTimeout = 20
	local waypointsFailed = 0
	local maxWaypointsFailed = 3
	local currentDist = math.huge
	local bestDist = math.huge
	local bestDistTimeout = 3 -- Seconds×××
```

```
	local velocityTimeout = 10
	local killPath = false

	local waypoints = findTurns(self, path)

	for i,waypoint in ipairs(waypoints) do

		local waypointProgressTimestamp = tick()

		if killPath or not self.Instance then
			break
		end

		local reversed = false

		waypointTimeout = (self×FrontBumper×WorldPosition-
waypoint×Position)×Magnitude/2
		waypointTimeout = math×clamp(waypointTimeout,100,1000)

		while not self:IsAtWaypoint(waypoint.Position) do

			reversed = false

			debugger.waypoint(waypoint.Position)

			-- If an obstacle blocks our path and slows our velocity to almost
a stop reverse.
			local velocity = self.Seat.Velocity.Magnitude
			if not reversed and (self.FrontBumper.WorldPosition -
waypoint.Position).Magnitude >= currentDist and velocity < 5 then
				if self:CheckBumper() then
					local desiredAngle =
core.findYaw(self.FrontBumper.WorldPosition,waypoint.Position,self.Occupant.R
oot)
					self:SmartReverse(waypoint.Position, -desiredAngle,
1)
					reversed = true
				end
			end

			local closer = false
			local newDist = (self×FrontBumper×WorldPosition -
waypoint×Position)×Magnitude
			if currentDist > newDist then
				closer = true
			end
```

```
                currentDist = newDist
                if velocity<1 then
                        velocityTimeout = velocityTimeout-1
                        if velocityTimeout<=0 then
                                velocityTimeout = 10
                                local desiredAngle =
core.findYaw(self.FrontBumper.WorldPosition,waypoint.Position,self.Occupant.R
oot)
                                self:SmartReverse(waypoint.Position, -desiredAngle,
1)
                                reversed = true
                                --self:Reverse(1.5)
                        end
                else
                        velocityTimeout = 10
                end

                --Ensure we are making progress to our end goal.
                if closer then
                        --bestDistTimeout = 10
                        waypointProgressTimestamp = tick()
                else
                        --bestDistTimeout = bestDistTimeout - 1
                        --if bestDistTimeout <= 0 then
                        if tick() - waypointProgressTimestamp > bestDistTimeout
then
                                killPath = true
                                pathFailures = pathFailures + 1
                                break
                        end
                end

                --Ensure we aren't stuck on the same waypoint for too long.
                --This may be redundant with the new end goal position timeout.
                if not closer then
                        waypointTimeout = waypointTimeout - 1
                end
                if waypointTimeout <= 0 then --and myRoot.Velocity.Magnitude
< 3
                        waypointsFailed = waypointsFailed + 1

                        if i == #waypoints or waypointsFailed >=
maxWaypointsFailed or not
core.checkDirectPath(self.Occupant,waypoints[i+1],self.Instance,self.FrontBum
per.WorldPosition) then
                                pathFailures = pathFailures + 1
                                killPath = true
```

```
                        break
                    end
                end

                -- Skip to next waypoint if we can see it and close enough to
current one.
                -- This results in very natural looking turns.
                if self.Seat.Velocity.Magnitude>50 and currentDist < 20 and i ~=
#waypoints and
core.checkDirectPath(self.Occupant,waypoints[i+1],self.Instance,self.RearBump
er.WorldPosition) then
                    break
                end

                if instance then
                    -- If we are driving to a target we need to do additional
checks.
                    if toTarget then
                        if not self.Occupant:CheckTarget() then
                            killPath = true
                            break
                        end

                        local dist =
core.checkDist(self.Occupant.Root,currentTarget)
                        if dist <
self.Occupant.Settings.Vehicle.MinDrivingDist.Value then

                            -- If I am alone and close enough the target then I
can stop if I can see.
                            -- If I have a gunner and I am close enough then I
can stop if the GUNNER can see.
                            if (self:CheckTurretSightToTarget() and
self.Instance:GetAttribute("Gunner"))
                                or (self.Occupant:CheckSightToTarget() and
not self.Instance:GetAttribute("Gunner"))then
                                killPath = true
                                break
                            end

                        end

                        if currentTarget ~= self.Occupant:GetTarget() then
                            killPath = true
                            break
                        end
                    else
```

```
                        -- If we are driving to an instance it is probably to pick
up the gunner.
                        local dist = core.checkDist(self.Body,instance)
                        if dist <=
self.Occupant.Settings.Vehicle.MinDrivingDist.Value * 0.75 then
                            killPath = true
                            break
                        end
                    end
                end

                -- TODO Idk if these values will update from outside of the
module
                if self.Occupant:GetTarget() and self.Occupant.GoingHome then
                    killPath = true
                    break
                end

                if not self:Validate() then
                    break
                end

                self:DriveTowards(waypoint.Position)

                task.wait()
            end

            if i == #waypoints and waypointsFailed == 0 then
                pathFailures = 0
            end
        end

        if self.Instance then
            self:Go(0)
        end
end

local VehicleBlacklist = {}
function Vehicle.isBlacklisted(npc,vehModel)
        for _, veh in pairs(VehicleBlacklist) do
            if veh.Instance == vehModel then --and npc:GetTarget() ==
veh.Target then
                return tick() - veh.Time <
npc.Settings.Vehicle.BlacklistDelay.Value
            end
        end
        return false
```

```
end

-- Prevent a Vehicle from being used for a certain target
function Vehicle:Blacklist(target)
	for i, veh in pairs(VehicleBlacklist) do
		if veh.Instance == self.Instance and veh.Target == target then
			VehicleBlacklist[i].Time = tick()
			return
		end
	end

	table.insert(VehicleBlacklist, {
		Time = tick(),
		Instance = self.Instance,
		Target = target
	})
end

function Vehicle:GetSpeed()
	return self.Seat.Velocity.Magnitude
end

-- True/False if we were able to get some kind of path going.
function Vehicle:DriveTo(pos,instance)

	if instance and instance == self.Occupant:GetTarget() then
		local points = self.Occupant:GetTargetPoints()
		for i,v in ipairs(points) do
			local vehPath = pathing.getVehiclePath(self.Instance, v, 20, true)
			if vehPath then
				self:DrivePath(vehPath, instance)
				return true
			end
		end
	end


	local vehPath = pathing.getVehiclePath(self.Instance, pos, 20)
	if vehPath then
		self:DrivePath(vehPath,instance)
		return true
	end
	return false
end

-- Returns true/false based on if the the turret would be able to shoot the target
function Vehicle:CheckTurretSightToTarget()
```

```
	local top = self.Instance.Turret.Top
	return core.isLine(top.Position,self.Occupant:GetTarget().Position,{
		self.Occupant:GetTarget().Parent,
		self.Instance,
		self.Occupant.Instance
	})
end


function Vehicle:GetAllies()
	local allies = {}
	if self.Instance:GetAttribute("Gunner") then
		local allyModels =
CollectionService:GetTagged(self.Occupant.Settings.Team.Value)
		for i,v in ipairs(allyModels) do
			local query = v:FindFirstChild("Query")
			if query then
				local allyNPC = query:Invoke()

				-- Find allies we need to pickup / wait for that aren't already
in the vehicle.
				if allyNPC.Vehicle and allyNPC×Vehicle×Instance ==
self.Instance and not allyNPC.Vehicle.InVehicle
					and allyNPC.Vehicle.Role ~= "Driver" then
					--return allyNPC
					table.insert(allies,allyNPC)
				end
			end
		end
	end
	return allies
end

-- Returns true/false if we need to pick up more allies.
function Vehicle:PickupNearestAlly()

	local allies = self:GetAllies()
	local ally

	-- Get the closest ally
	local dist = math.huge
	for _,tempAlly in allies do
		local tempDist = core×checkDist(tempAlly×Root,self.Occupant.Root)
		if tempDist < dist then
			dist = tempDist
			ally = tempAlly
		end
```

```
	end

	-- No allies to pickup
	if not ally then return end

	if dist > 20 then
		self:DriveTo(ally.Root.Position, ally.Root)
	end

	return true
end

function Vehicle:SetTeam(team: string)
	self.Team.Value = team
end

-- Checks if the vehicle's team is none or ally.
function Vehicle.checkTeam(npc: ObjectValue, vehicleModel: Model)
	local team = vehicleModel.Settings.Team.Value
	return team == npc.Team or team == ""
end

return Vehicle
local targeting = {}

local targetings = script.Parent
local core = require(targetings×Core)

function targeting:GetSeeTarget()
	local dist = self.Settings.DetectionRange.Value
	local target = nil
	local seeTargets = {}

	for _,v in ipairs(workspace:GetChildren()) do
		local human = v:FindFirstChild("Humanoid")
		local torso = v:FindFirstChild("HumanoidRootPart") or
v:FindFirstChild("Torso")
		if human and torso and v ~= self.Instance then
			if (self.Root.Position - torso.Position).Magnitude < dist and
human.Health > 0 and self:CheckSight(torso) and not core.isAlly(v) then
				return true
			end
		end
	end
	return false
end
```

```
function targeting:FindTarget()
	local dist = self.Settings.DetectionRange.Value
	local target = nil
	local activeAllies = {}
	local potentialTargets = {}

	local seeTargets = {}
	for _,v in ipairs(workspace:GetChildren()) do
		local human = v:FindFirstChild("Humanoid")
		local torso = v:FindFirstChild("HumanoidRootPart") or
v:FindFirstChild("Torso")
		if human and torso and v ~= self.Instance then
			if (self.Root.Position - torso.Position).Magnitude < dist and
human.Health > 0 then
				local teammate = core×isAlly(v)
				if teammate then
					table.insert(activeAllies,torso)
				else
					table.insert(potentialTargets,torso)
				end
			end
		end
	end

	self:SetPotentialTargets(potentialTargets)
	self:SetActiveAllies(activeAllies)

	if #potentialTargets > 0 then
		for i,v in ipairs(potentialTargets) do
			if self:CheckSight(v) then
				table.insert(seeTargets,v)
			end
		end
		if #seeTargets > 0 then
			for i,v in ipairs(seeTargets) do
				local tempDist = (self×Root×Position -
v×Position)×Magnitude
				if tempDist < dist then
					target = v
					dist = tempDist
				end
			end
		else
			for i,v in ipairs(potentialTargets) do
				local tempDist = (self×Root×Position -
v×Position)×Magnitude
```

```
                        if tempDist < dist then
                            target = v
                            dist = tempDist
                        end
                    end
                end
            end
            self:SetTarget(target)
end

function targeting:GetTarget()
            return self.Target.Instance
end

function targeting:SetTarget(target)
            self.Target.Instance = target
            --self×Target×Vehicle = nil
end

function targeting:GetTargetVehicle()
            return self.Target.Vehicle
end

function targeting:SetTargetVehicle(vehicle: Instance)
            self×Target×Vehicle = vehicle
end

function targeting:SetTargetLastSeen()
            self.Target.LastSeen = tick()
end

-- Determines if our current target is valid
function targeting:CheckTarget()
            local target = self.Target.Instance
            if target and target.Parent and target.Parent.Humanoid.Health>0 then
                        return true
            end
            return false
end

function targeting:GetTargetLastSeen()
            return self.Target.LastSeen
end

local raycastParams = RaycastParams×new()
function targeting:CheckSightToTarget(dist: IntValue)
            if not self:GetTarget() then
```

```
		return false
	end
	if not dist then
		dist = self×Settings×M4.Range×Value
	end

	local ignoreList = {self.Instance}
	if self.Vehicle and self.Vehicle:IsInVehicle() then
		table.insert(ignoreList, self.Vehicle.Instance)
	end

	local enemyTargetVehicle = self:GetTargetVehicle()
	if enemyTargetVehicle then
		table.insert(ignoreList, enemyTargetVehicle)
	end

	raycastParams×FilterDescendantsInstances = ignoreList
	local result = workspace:Raycast(self.Head.Position,
(self:GetTarget().Position - self.Head.Position).Unit*dist,raycastParams)
	if result and result.Instance.Parent then
		if result.Instance:IsDescendantOf(self:GetTarget().Parent) or
result.Instance.Name == "Bullet" then
			return true
		else
			local human = core×getHuman(result×Instance×Parent)
			if human and human.Health>0 and not core.isAlly(human.Parent)
and human.RootPart then
				self:SetTarget(human.RootPart)
				return true
			end
		end
	end
	return false
end

-- Generates 8 points around the target for us to potentially drive to.
function targeting:GetTargetPoints()
	local points = 8
	local radius = self×Settings×Vehicle×MinDrivingDist×Value × 0.75
	local drivablePoints = {}
	for i = 1, points do
		local angle = i × (2 × math×pi  / points)
		local x = math×cos(angle) × radius
		local z = math×sin(angle) × radius
		local position = self:GetTarget().Position + Vector3.new(x,3,z)

		local result = workspace:Raycast(position, Vector3.new(0,-10,0))
```

```
            if core.isLine(position, self:GetTarget().Position,
{self.Instance,self:GetTarget().Parent})
                    and result then

                    -- Update so that the point is near the ground so pathfinding will
work
                    position = Vector3.new(position×X, result.Position.Y + 0.5,
position.Z)

                    table.insert(drivablePoints, position)
            end
        end

        return drivablePoints
end

function targeting:GetIncomingAttackers()

        local targets = {}

        local params = RaycastParams×new()
        params×FilterDescendantsInstances = {self.Instance}
        for i = -20, 20, 2 do
            local origin = self×Root×Position + Vector3.new(-20,0,i)
            local dir = Vector3.new(40,0,0)
            local result = workspace:Raycast(origin, dir)
            if result then
                local human = core×getHuman(result×Instance×Parent)
                local root =
result.Instance.Parent:FindFirstChild("HumanoidRootPart")
                if human and human.Health>0 and root and not
core.isAlly(result.Instance.Parent) then
                    if self:CheckSight(root) then
                        table.insert(targets,root)
                    end
                end
            end
        end

        --Sort through targets to target the closest one to us
        local dist = math.huge
        local tempTarget = nil
        for _,potentialTarget in ipairs(targets) do
            if core.checkDist(potentialTarget,self.Root) < dist then
                dist = core.checkDist(potentialTarget,self.Root)
                tempTarget = potentialTarget
```

```
		end
	end

	if tempTarget then
		self:SetTarget(tempTarget)
	end
end

return targeting
local pathing = {}

local debugger = require(script×Parent×Debug)

local pathFinding = game:GetService("PathfindingService")
function pathing.generatePath(startPos,goal,vehicle)

	local path
	if vehicle then

		local size = vehicle:GetExtentsSize()
		local radius = size×X

		local pathParams = {
			AgentRadius = radius,
			AgentHeight = size×Y,
			AgentCanJump = false
		}
		path = pathFinding:CreatePath(pathParams)
	else
		path = pathFinding:CreatePath()
	end

	path:ComputeAsync(startPos,goal)

	return path

end

function pathing.getVehiclePath(vehicle: Instance, pos: Vector3,
minDrivingDist: IntValue, noStubs: BoolValue)

	--local veh = npc×Vehicle

	local body = vehicle×Body×Body
	local frontBumper = body×FrontBumper
	local rearBumper = body×RearBumper
```

```
	local startPos = frontBumper×WorldPosition +
frontBumper×WorldCFrame×LookVector

	-- For loop for checking paths from the front bumper and rear bumper.
	for _=1, 2 do

		local mark = debugger×mark(startPos)
		--mark×BrickColor = BrickColor.new("Pink")

		local path = pathing×generatePath(startPos,pos,vehicle)
		if path×Status == Enum.PathStatus.Success then
			--This path will take us directly to the target
			return path
		elseif not noStubs then
			--Generate walking path and see how much of it is drivable
			local tempPath = pathing×generatePath(startPos,pos,false)
			debugger.visualizePath(tempPath)
			if tempPath×Status == Enum.PathStatus.Success then
				local tempWaypoints = tempPath:GetWaypoints()
				for index = 1, 5 do

					local targetWaypoint = math.ceil(#tempWaypoints -
#tempWaypoints*(index/6))


					--Make sure path contains at least 10 waypoints
					if targetWaypoint > 10 then --
mySettings.MinDrivingDist.Value
						local waypointPos =
tempWaypoints[targetWaypoint].Position

						local mark = debugger×mark(waypointPos)

						--Ensure path length meets our minimum driving
distance requirement
						--If path is too short we should just walk it
						if (startPos - waypointPos).Magnitude >
minDrivingDist then
							--if core.checkDist(startPos,waypointPos) >
minDrivingDist then
							local path =
pathing×generatePath(startPos,waypointPos,vehicle)
							if path×Status == Enum.PathStatus.Success
then
								debugger.visualizePath(path)
								return path
```

```
                                    end
                                else
                                    break
                                end
                            end

                        end
                    end
                end

                startPos = rearBumper×WorldPosition +
rearBumper×WorldCFrame×LookVector×2
        end
end

return pathing
local Vehicle = require(script×Parent×Vehicle)

local Passenger = {}
Passenger.__index = Passenger
setmetatable(Passenger, Vehicle)

-- Constructor
function Passenger.new(vehicle, npc)

        local o = {}
        o.Instance = vehicle
        o.Occupant = npc
        o.Seat = vehicle.Seat
        o.Role = "Passenger"
        o.InVehicle = false
        o×Body = vehicle.Body.Body
        o.Door = vehicle.Body.Body.SideDoor
        o.Team = vehicle.Settings.Team

        vehicle:SetAttribute(o.Role, true)

        setmetatable(o, Passenger)

        o:SetTeam(npc.Team)

        return o
end

function Passenger:Enter()
        self.Seat:Sit(self.Occupant.Human)
        self.Occupant:YieldWeapons()
```

```
	self.InVehicle = true
end

function Passenger:Exit()
	self×Occupant×Human×Sit = false
	if not self.Occupant:IsDead() then
		task.wait()
		self.Occupant.Root.CFrame = self.Door.WorldCFrame *
CFrame.new(1.5,0.5,0)
	end
	self.Occupant:SetVehicleTimestamp()
	self:Destroy()
	return nil
end



return Passenger
local NPC = {}
NPC.__index = NPC

local Core = require(script×Parent×Core)

function NPC.new(o)

	if not o.Instance then
		error("No instance provided for the NPC.")
		return
	end

	setmetatable(o, NPC)

	-- NPC instance parts
	o×Root = o.Instance.HumanoidRootPart
	o×Human = o.Instance.Humanoid
	o.Head = o.Instance.Head
	o.Torso = o.Instance.Torso
	o.Neck = o.Torso.Neck
	o.LeftArm = o.Instance["Left Arm"]
	o.RightArm = o.Instance["Right Arm"]
	o.LeftLeg = o.Instance["Left Leg"]
	o.RightLeg = o.Instance["Right Leg"]
	o.LeftShoulder = o.Torso["Left Shoulder"]
	o.RightShoulder = o.Torso["Right Shoulder"]
	o.LeftHip = o.Torso["Left Hip"]
	o.RightHip = o.Torso["Right Hip"]

	o.Settings = o.Instance.Settings
```

```
	o.Knife = o.Instance.Knife
	o.StabAnimation = o.Instance.Stab
	o.StabPunchAnimation = o.Instance.StabPunch

	-- Shoulder hinges for when in the gunner or driver positions
	o.ShoulderHingeR = o.Torso.ShoulderHingeR
	o.ShoulderHingeL = o.Torso.ShoulderHingeL

	o×M4 = o.Instance.M4
	o.Grenade = o.Instance.Grenade
	o.BodyRotate = o.Root.AlignOrientation
	o.Dead = false
	o.Target = {}
	o×Target×Root = nil
	o.Target.LastSeen = tick()
	o.GoingHome = false
	o×Vehicle = nil
	o.Team = o.Settings.Team.Value
	o×M4Equipped = false
	o×M4Lowered = false
	o×M4Aimed = false
	o.Reloading = false
	o.WeaponCool = true
	o.KnifeEquipped = false
	o.GrenadeCool = true
	o.Mood = "Idle"
	o.Face = o.Head.faceIdle
	o.Faces = o.Head.Faces
	o.TookDamage = false
	o×PotentialTargets = {}
	o×ActiveAllies = {}
	o.Mag = o.Settings.M4.MagSize.Value

	return o
end

-- Allows all functions from the module to be natively called from the NPC
object.
function NPC:Inherit(module: Table)
	for k,v in pairs(module) do
		if not self[k] then
			self[k] = v
		else
			warn("Tried to inherit an already existing function ", k, " from ",
module)
		end
	end
```

```
end

function NPC:IsDead()
	return self.Human.Health <= 0
end

function NPC:SetMag(mag: IntValue)
	self.Mag = mag
end

function NPC:GetMag()
	return self.Mag
end

function NPC:SetActiveAllies(allies: Table)
	self×ActiveAllies = allies
end

function NPC:GetActiveAllies()
	return self.ActiveAllies
end

function NPC:SetPotentialTargets(targets: Table)
	self×PotentialTargets = targets
end

function NPC:GetPotentialTarget()
	return self.PotentialTargets
end

function NPC:SetTookDamage(damageTaken)
	self.TookDamage = damageTaken
end

function NPC:GetTookDamage()
	return self.TookDamage
end

function NPC:SetFace(face: Decal)
	self.Face = face
end

function NPC:GetFace()
	return self.Face
end

function NPC:SetMood(mood: string)
```

```
	self.Mood = mood
end

function NPC:GetMood()
	return self.Mood
end

function NPC:SetGrenadeCool(cool: BoolValue)
	self.GrenadeCool = cool
end

function NPC:GetGrenadeCool()
	return self.GrenadeCool
end

function NPC:SetKnifeEquipped(equipped)
	self.KnifeEquipped = equipped
end

function NPC:GetKnifeEquipped()
	return self.KnifeEquipped
end

function NPC:SetWeaponCool(cool)
	self.WeaponCool = cool
end

function NPC:GetWeaponCool()
	return self.WeaponCool
end

function NPC:SetReloading(reloading: BoolValue)
	self.Reloading = reloading
end

function NPC:GetReloading()
	return self.Reloading
end

function NPC:SetM4Aimed(aimed: BoolValue)
	self×M4Aimed = aimed
end

function NPC:GetM4Aimed()
	return self.M4Aimed
end
```

```
function NPC:SetM4Lowered(lowered: BoolValue)
	self×M4Lowered = lowered
end

function NPC:GetM4Lowered()
	return self.M4Lowered
end

function NPC:SetM4Equipped(equip: BoolValue)
	self×M4Equipped = equip
end

function NPC:GetM4Equipped()
	return self.M4Equipped
end

-- Keeps track of when we left a vehicle.
function NPC:SetVehicleTimestamp()
	self.LastEnteredVehicle = Core.tick()
end

-- Ensures we don't enter / exit a vehicle too quickly.
function NPC:CanEnterVehicle()
	return not self.LastEnteredVehicle or Core.tick() - self.LastEnteredVehicle
> self.Settings.Vehicle.Cooldown.Value
end

function NPC:CanExitVehicle()
	return not self.LastEnteredVehicle or Core.tick() - self.LastEnteredVehicle
> self.Settings.Vehicle.Cooldown.Value
end

function NPC:CheckSight(target)

	local ignoreList = {self.Instance}
	if self.Vehicle and self.Vehicle:IsInVehicle() then
		table.insert(ignoreList,self.Vehicle.Instance)
	end

	local enemyTargetVehicle = self:GetTargetVehicle()
	if enemyTargetVehicle then
		table.insert(ignoreList, enemyTargetVehicle)
	end

	local raycastParams = RaycastParams×new()
	raycastParams×FilterDescendantsInstances = ignoreList
```

```
	local result = workspace:Raycast(self.Head.Position,(target.Position -
self.Head.Position).Unit*self.Settings.M4.Range.Value,raycastParams)
	if result and result.Instance.Parent then
		if result.Instance:IsDescendantOf(target.Parent) or
result.Instance.Name == "Bullet" then
			self:SetTargetLastSeen()
			return true
		else
			local human = Core×getHuman(result×Instance×Parent)
			if human and human.Health>0 and not
Core.isAlly(human.Parent) and human.RootPart then
				self:SetTarget(human.RootPart)
				self:SetTargetLastSeen()
				return true
			end
		end
	end
	return false
end

function NPC:CheckShot()
	local raycastParams = RaycastParams×new()
	local ignoreList = {self.Instance}

	local enemyTargetVehicle = self:GetTargetVehicle()
	if enemyTargetVehicle then
		table.insert(ignoreList, enemyTargetVehicle)
	end

	raycastParams×FilterDescendantsInstances = ignoreList



	local dist = Core.checkDist(self.Root,self:GetTarget())
	local result1 = workspace:Raycast(self.M4.Barrel.WorldPosition,
(self:GetTarget().Position -
self.M4.Barrel.WorldPosition).Unit*dist,raycastParams)
	local result2 = workspace:Raycast(self.Head.Position,
(self:GetTarget().Position - self.Head.Position).Unit*dist,raycastParams)
	if result1 and result2 and result1.Instance×Parent ==
result2.Instance.Parent then
		if result1.Instance:IsDescendantOf(self:GetTarget().Parent) or
result1.Instance×Name == "Bullet" then
			return true
		else
			local human =
```

```
result1.Instance×Parent:FindFirstChild("Humanoid")
                    if human and human.Health>0 and not
Core.isAlly(human.Parent) then
                            self:SetTarget(human.RootPart)
                            return true
                    end
            end
        end
        return false
end

function NPC:FacingTarget()
        --Make sure we're facing the target
        local currentLookVector = self×Root×CFrame×LookVector
        local targetPos = self:GetTarget().Position
        local goalLookVector = CFrame×new(self×Root×Position,Vector3.new(
                targetPos.X,
                self.Root.Position.Y,
                targetPos.Z
                )).LookVector
        local difference = (currentLookVector - goalLookVector)×Magnitude
        if difference < 0.5 then
                return true
        else
                return false
        end
end

return NPC
local movement = {}

--Services
local pathfindingService = game:GetService("PathfindingService")
local runService = game:GetService("RunService")

local modules = script.Parent
local core = require(modules×Core)
local debugger = require(modules.Debug)

local marine = script.Parent.Parent.Parent
local myHuman = marine.Humanoid
local myRoot = marine.HumanoidRootPart



local moveEvent = Instance.new("BindableEvent")
function movement.moveToFinished()
        local success = true
```

```
		local completed = false

	core.spawn(function()
		task.wait(1)
			if not completed then
			success = false
			moveEvent:Fire()
			end
	end)

	core.spawn(function()
		myHuman.MoveToFinished:Wait()
		moveEvent:Fire()
	end)

		moveEvent.Event:Wait()

		completed = true

	return success
end

local path = pathfindingService:CreatePath()
function movement:PathToLocation(target)
		--if not target or not target.Parent then return end
		if not target or (typeof(target) == "Instance" and not target.Parent) then
return end

		local pathingToTarget = false
		if target == self:GetTarget() then
			pathingToTarget = true
		end

		-- Added so that we can path to attachments which specific spots on a
vehicle to enter.
		local posProperty = "Position"
		if target:IsA("Attachment") then
			posProperty = "WorldPosition"
		end

		path:ComputeAsync(myRoot.Position, target[posProperty])

		if path.Status == Enum.PathStatus.NoPath then
			return false
		end

		local waypoints = path:GetWaypoints()
```

```
	--Make sure that path is actually valid
	if waypoints[1] and core.checkDist(waypoints[1],myRoot) > 10 then
		task.wait()
		self:PathToLocation(target)
	end

	debugger.visualizePath(path)

	local canSee = self:CheckSightToTarget()


	for _,waypoint in ipairs(waypoints) do
		if waypoint.Action == Enum.PathWaypointAction.Jump and (not
self.Vehicle or not self.Vehicle:IsInVehicle()) then
			self×Human×Jump = true
		end
		self.Human:MoveTo(waypoint.Position)
		task.spawn(function()
			local tries = 0
			while self.Human.WalkToPoint.Y > self.Root.Position.Y - 1 and
(not self.Vehicle or not self.Vehicle:IsInVehicle())
				and myHuman.Health>0 do

				tries += 1
				myHuman.Jump = true
				task.wait(0.1)
				if tries > 10 then
					break
				end
			end
		end)

		-- Bunch of logic to determine when the path should be interrupted.
		local moveSuccess = movement×moveToFinished()
		local dist = core.checkDist(self.Root,target[posProperty])

		-- Sometimes the NPC will get stuck behind it's vehicle or other
unanchored objects.
		-- I think a simple jump could fix this.
		if core.raycast(self.Root.Position, (waypoint.Position -
self.Root.Position).Unit * 2, {self.Instance}) then
			if not self.Vehicle or not self.Vehicle:IsInVehicle() then
				self×Human×Jump = true
				task.wait(1.2)
			end
		end
```

```
            if not moveSuccess then
                break

            elseif core.checkDist(target[posProperty],waypoints[#waypoints]) >
30 then
                -- Kind of an exception for when pathing to an attachment which
is probably a vehicle
                -- Try to path to it again.
                self:PathToLocation(target)
                break

            elseif not target.Parent then
                break

            elseif pathingToTarget then

                -- Additional checks when we are pathing to a target.
                if core.runAtTS("PathToLocationCheckSight",1) then
                    canSee = self:CheckSightToTarget()
                end

                if dist < 30 and canSee then
                    break

                elseif (canSee and target.Position.Y <= myRoot.Position.Y-5)
then
                    self.Human:MoveTo(myRoot.Position)
                    break
                elseif target ~= self:GetTarget() or not self:CheckTarget() then
                    break
                end

            elseif target:IsA("Attachment") and dist <
self.Settings.Vehicle.EnterDist.Value then
                break
            end
        end

        return true
end

function movement:WalkRandom()
        local randX = math×random(-100,100)
        local randZ = math.random(-100,100)
        local goal = self.Root.Position + Vector3.new(randX, 0, randZ)
```

```
	local path = pathfindingService:CreatePath()
	path:ComputeAsync(self.Root.Position, goal)
	local waypoints = path:GetWaypoints()

	if path.Status == Enum.PathStatus.Success then
		for i,waypoint in ipairs(waypoints) do
			if waypoint.Action == Enum.PathWaypointAction.Jump then
				self×Human×Jump = true
			end
			self.Human:MoveTo(waypoint.Position)
			core.spawn(function()
				task.wait(0.5)
				if self.Human.WalkToPoint.Y > self.Root.Position.Y then
					self×Human×Jump = true
				end
			end)
			local moveSuccess = myHuman.MoveToFinished:Wait()
			if not moveSuccess or self:GetTarget() then
				break
			end
		end
	else
		task.wait(2)
	end
end

--When strafing make sure you can see target from end pos
function movement:Strafe()
	local goalPos = self:GetTarget().Position
	local origin,dir = self.Root.Position,nil

	local action = math.random(2)
	if action == 1 then
		dir = self.Root.CFrame.RightVector
	elseif action == 2 then
		dir = -self×Root×CFrame×RightVector
	end

	for _=1,10 do
		if not core.raycast(origin,dir*2) and
core.raycast(origin,Vector3.new(0,-7,0)) then
			origin += dir*2
		else
			origin += dir*-2
			break
		end
	end
```

```
	local _,result = core.raycast(origin,(self:GetTarget().Position - origin).Unit *
30)
	if result then
		if result.Instance:IsDescendantOf(self:GetTarget().Parent) then

		else

			origin = self.Root.Position
		end
		origin = self.Root.Position
	end

	self.Human:MoveTo(origin)
end

function checkDirection(npc, dir)
	local goalPos = npc:GetTarget().Position
	local origin = myRoot×Position

	local success = false
	for i=1, 5 do
		if not core.raycast(origin,dir*1) and
core.raycast(origin,Vector3.new(0,-7,0)) then
			origin = origin + dir×1
		else
			origin = origin + dir×-1
			break
		end

		if i == 5 then
			success = true
		end
	end

	return success,origin
end

function movement:Retreat()

	--Check if we can retreat directly from the enemy.
	local result,pos = checkDirection(self,-(self:GetTarget().Position -
self.Root.Position).Unit)
	if result then
		self.Human:MoveTo(pos)
		return
	end
```

```
	--Check if we can move away to the right.
	result,pos = checkDirection(self,self.Root.CFrame.RightVector)
	if result then
		self.Human:MoveTo(pos)
		return
	end

	--Check if can move away to the left.
	result,pos = checkDirection(self,-self×Root×CFrame×RightVector)
	if result then
		self.Human:MoveTo(pos)
		return
	end
end

function movement:SlowAdvance()
	local goalPos = self:GetTarget().Position
	local origin = self×Root×Position

	local maxIterations = 30
	repeat
		maxIterations -= 1
		if not core.raycast(origin,(goalPos - origin).Unit*1) and
core.raycast(origin,Vector3.new(0,-7,0)) then
			origin = origin + (goalPos - origin)×Unit×1
		else
			origin = origin + (goalPos - origin)×Unit×-1
			break
		end
		if maxIterations <= 0 then
			break
		end
	until false

	self.Human:MoveTo(origin)
end

function movement:ChaseTargetDirectly()
	self.Human:MoveTo(self:GetTarget().Position)
end

return movement
local Gunner = {}
Gunner.__index = Gunner

local Vehicle = require(script×Parent×Vehicle)
```

```
local core = require(script×Parent×Core)
local combat = require(script×Parent×Combat)
local debugger = require(script.Parent.Debug)
local formulas = require(script×Parent×Formulas)

setmetatable(Gunner, Vehicle)

local DebrisService = game:GetService("Debris")
local CollectionService = game:GetService("CollectionService")
local PhysicsService = game:GetService("PhysicsService")
local RunService = game:GetService("RunService")


local MARINE_VEHICLE_TAG = "MarineVehicle"

function Gunner.new(vehicle, npc)

	local o = {}
	o.Instance = vehicle
	o.Seat = vehicle.VehicleSeat
	o.Role = "Gunner"
	o.BarrelMotor = vehicle.Turret.Barrel.BarrelMotor
	o.Aimer = vehicle.Body.Body.Aimer
	o.Aiming = false
	o.Shooting = false
	o.ShootingRequestStop = false
	o.Top = vehicle.Turret.Top
	o.Barrel = vehicle.Turret.Barrel
	o×Body = vehicle.Body.Body
	o.Occupant = npc
	o.InVehicle = false
	o.TopToBaseHinge = o.Top.TopToBaseHinge
	o.Team = vehicle.Settings.Team

	setmetatable(o,Gunner)

	o:SetTeam(npc.Team)

	vehicle:SetAttribute(o.Role, true)

	return o
end

-- Teleport NPC into the Gunner position
function Gunner:Enter()

	local npc = self.Occupant
```

```
	npc:YieldWeapons()

	npc×Human×PlatformStand = true
	npc.Root.CFrame = self.Instance.Turret.Base.CFrame *
CFrame.new(0,1.5,1.7)

	self.GunnerWeld = Instance.new("WeldConstraint")
	self×GunnerWeld×Part0 = self.Instance.Turret.Base
	self×GunnerWeld×Part1 = npc.Root
	self.GunnerWeld.Parent = npc.Root
	self×Top×AlignOrientation×Attachment1 = self.Aimer

	for i,v in ipairs(npc.Instance:GetDescendants()) do
		if v:IsA("BasePart") then
			PhysicsService:SetPartCollisionGroup(v,"Passengers")
		end
	end

	RunService.Heartbeat:Wait()
	RunService.Heartbeat:Wait()

	-- Setup NPC stance
	npc×LeftShoulder.Enabled = false
	npc×RightShoulder.Enabled = false
	npc×LeftHip.Enabled = false
	npc×RightHip.Enabled = false

	-- Set CFrame
	npc.RightLeg.CFrame = npc.Root.CFrame * CFrame.new(0.6,-2,0) *
CFrame.fromEulerAnglesXYZ(0,0,math.rad(8))
	npc.LeftLeg.CFrame = npc.Root.CFrame * CFrame.new(-0.6,-2,0) *
CFrame.fromEulerAnglesXYZ(0,0,math.rad(-8))

	npc.Instance.Torso.GunnerWeld_LL.Enabled = true
	npc.Instance.Torso.GunnerWeld_RL.Enabled = true

	npc×LeftShoulder.Enabled = false
	npc×RightShoulder.Enabled = false
	npc×ShoulderHingeR.Enabled = true
	npc×ShoulderHingeL.Enabled = true

	self.InVehicle = true
end

function Gunner:Exit()

	local npc = self.Occupant
```

```
	-- Reset the turret back to the original orientation
	self.Aimer.WorldCFrame = self.Body.CFrame

	npc×ShoulderHingeR.Enabled = false
	npc×ShoulderHingeL.Enabled = false
	npc.Instance.Torso.GunnerWeld_LL.Enabled = false
	npc.Instance.Torso.GunnerWeld_RL.Enabled = false

	--npc.Aimer.Parent = npc.Vehicle.Body
	npc.Aimer = nil
	if self.GunnerWeld then
		self.GunnerWeld:Destroy()
	end
	self.GunnerWeld = nil

	for i,v in ipairs(npc.Instance:GetDescendants()) do
		if v:IsA("BasePart") then
			PhysicsService:SetPartCollisionGroup(v, "Default")
		end
	end

	-- Setup NPC stance
	npc×LeftShoulder.Enabled = true
	npc×RightShoulder.Enabled = true
	npc×LeftHip.Enabled = true
	npc×RightHip.Enabled = true

	RunService.Heartbeat:Wait()

	npc.Root.CFrame = self.Instance.Body.Body.RearBumper.WorldCFrame *
CFrame.new(0,0,-3)
	npc×Human×PlatformStand = false

	npc:SetVehicleTimestamp()

	self:Destroy()

	return nil
end

local flashIndex = 1
function Gunner:StartShootingTurret()
	if self.Shooting then return end
	self.Shooting = true
	self.ShootingRequestStop = false
	self.BarrelMotor.AngularVelocity = 50
```

```
	local cooldown = 0.4
	task.spawn(function()
		repeat
			local shootSound = self.Barrel.Shoot:Clone()
			shootSound.Parent = self.Barrel
			shootSound:Play()
			DebrisService:AddItem(shootSound, shootSound.TimeLength)


			local bullet = combat×getBullet()
			RunService.Heartbeat:Wait()
			bullet×CFrame = self.Barrel.CFrame * CFrame.new(0,0,-2.3)
			RunService.Heartbeat:Wait() --Give bullet time to render from
barrel
			combat.pushBullet(self.Occupant,bullet)

			local flashCoroutine = coroutine.create(function()
				self.Barrel.Tip.MuzzleFlashEffect:Emit(15)
				self.Barrel.Tip.MuzzleFlashInnerEffect:Emit(15)
				self.Barrel.Tip.SmokeEffect:Emit(3)
				self.Barrel.Tip.SparkEffect:Emit(20)
				self×Barrel×Tip×MuzzleDynamicLight.Enabled = true
				wait(0.05)
				self×Barrel×Tip×MuzzleDynamicLight.Enabled = false
			end)

			coroutine.resume(flashCoroutine)

			task.wait(cooldown)
			if cooldown > 0.1 then
				cooldown = cooldown - cooldown * 0.2
			end
		until not self.Shooting or self.ShootingRequestStop or
self.Occupant:IsDead() or not Vehicle.isValid(self.Occupant,self.Instance) or
			not self.Instance:GetAttribute("Gunner")

		self.Shooting = false
		self.BarrelMotor.AngularVelocity = 0
	end)
end

function Gunner:StopShootingTurret()
	self.ShootingRequestStop = true
end

--local function calcOffset(vehicle, offset)
```

```
--	-- Add inaccuracy
--	local offset = math×random(0, 100 -
vehicle×Occupant×Settings×Vehicle×Accuracy×Value) / 20

--	if math×random(2) == 1 then
--		offset = -offset
--	end

--	return offset
--end

function Gunner:StartAimingTurret()
	if self.Aiming then return end
	self.Aiming = true
	local offsetX = 0
	local offsetY = 0
	task.spawn(function()
		repeat
			local target = self.Occupant:GetTarget()
			if self.Occupant:CheckTarget() then

				-- Should hopefully improve performance since this should
60x less raycasts per second lol.

				if core.runAtTS("GunnerCheckSight",2) then
					if not
self:CheckSightTurret(self.Occupant.Settings.Vehicle.MaxTurretRange.Value)
then
						break
					end
				end

				-- Stuff to account for when aiming.
				-- The distance, our velocity, the target's velocity.
				local lookPos =
formulas×getAimVector3(self×Occupant×Root, target, 240) --120

				-- Simulates variations in the aim like a real person would
have.
				if core.runAtTS("GunnerOffset",0.5) then
					offsetX =
core×calcRandomOffset(self×Occupant×Settings×Vehicle×Accuracy×Value)
					offsetY =
core×calcRandomOffset(self×Occupant×Settings×Vehicle×Accuracy×Value)
				end

				lookPos = lookPos + Vector3.new(offsetX, offsetY, 0)
```

```
                        self.Aimer.WorldCFrame =
CFrame.lookAt(self.Instance.Turret.Top.Anchor.WorldPosition,
                                lookPos,
                                self.Body.CFrame.UpVector)
                    elseif tick() - self.Occupant:GetTargetLastSeen() > 2 then
                        self.Aimer.WorldCFrame = self.Body.CFrame
                    end

                    self.Occupant.ShoulderHingeR.TargetAngle =
-self.TopToBaseHinge.CurrentAngle
                    self.Occupant.ShoulderHingeL.TargetAngle =
-self.TopToBaseHinge.CurrentAngle

                    task.wait()
                until not self.Aiming or self.Occupant:IsDead() or not self.Instance or
not Vehicle.isValid(self.Occupant,self.Instance) or
                    not self.Instance:GetAttribute("Gunner")

                self.Aiming = false
            end)
end

function Gunner:StopAimingTurret()
        self.Aiming = false
end

function Gunner:Validate()
        if not Vehicle.isValid(self.Occupant,self.Instance) then
                self:Exit()
                return false
        end
        return true
end


function Gunner.IsValid(npc,veh)
        -- TODO: Add more stuff that makes it invalid
        return not veh:GetAttribute("Gunner")
end

function Gunner:CheckSightTurret()
        local dist = core.checkDist(self.Occupant:GetTarget(), self.Occupant.Root)

        local ignoreList = {self.Occupant.Instance, self.Instance}

        local enemyTargetVehicle = self.Occupant:GetTargetVehicle()
```

```
	if enemyTargetVehicle then
		table.insert(ignoreList, enemyTargetVehicle)
	end

	local raycastParams = RaycastParams×new()
	raycastParams×FilterDescendantsInstances = ignoreList
	local result = workspace:Raycast(self.Occupant.Head.Position,
(self.Occupant:GetTarget().Position - self.Occupant.Head.Position).Unit*dist,
raycastParams)
	if result and result.Instance.Parent then
		if result.Instance:IsDescendantOf(self.Occupant:GetTarget().Parent)
or result.Instance.Name == "Bullet" then
			self.Occupant:SetTargetLastSeen()
			return true
		end
		local human = core×getHuman(result×Instance×Parent)
		if human and human.Health>0 and not core.isAlly(human.Parent) then
			self.Occupant:SetTarget(human.RootPart)
			self.Occupant:SetTargetLastSeen()
			return true
		end
	end
	return false
end

function Gunner:CheckAim()
	return (self.Top.Anchor.WorldOrientation -
self.Aimer.WorldOrientation).Magnitude < 20
end

function Gunner:IsTargetInRange()
	if not self.Occupant:CheckTarget() then return false end
	return (self.Occupant.Root.Position -
self.Occupant:GetTarget().Position).Magnitude <=
self.Occupant.Settings.Vehicle.MaxTurretRange.Value
end

return Gunner
local Formulas = {}

function Formulas.getAimVector3(myRoot, target, bulletVel)

	local dist = (myRoot×Position - target×Position)×Magnitude

	--local toTarget = target×Position - (myRoot×Position + myRoot×Velocity ×
dist/15)
	local toTarget = target×Position - myRoot×Position
```

```
	local a = target.Velocity:Dot(target.Velocity) - (bulletVel * bulletVel)
	local b = 2 × target×Velocity:Dot(toTarget)
	local c = toTarget:Dot(toTarget)

	local p = -b / (2 × a)
	local q = math.sqrt((b * b) - 4 * a * c) / (2 * a)

	local t1 = p - q
	local t2 = p + 1
	local t

	if t1 > t2 and t2 > 0 then
		t = t2
	else
		t = t1
	end

	local aimSpot = target×Position + target×Velocity × t

	if aimSpot×X == "nan" then
		return Vector3.new(0,0,0)
	end

	return aimSpot
end

local velocity = 85


function computeLaunchAngle(dx,dy,grav)
	local g = math.abs(grav)
	local inRoot = (velocity×velocity×velocity×velocity) - (g × ((g×dx×dx) +
(2×dy×velocity×velocity)))
	if inRoot <= 0 then
		return .25 * math.pi
	end
	local root = math×sqrt(inRoot)
	local inATan1 = ((velocity*velocity) + root) / (g*dx)

	local inATan2 = ((velocity*velocity) - root) / (g*dx)
	local answer1 = math.atan(inATan1)
	local answer2 = math.atan(inATan2)
	if answer1 < answer2 then return answer1 end
	return answer2
end
```

```
function computeDirection(vec)
	local lenSquared = vec.magnitude * vec.magnitude
	local invSqrt = 1 / math.sqrt(lenSquared)
	return Vector3.new(vec.x * invSqrt, vec.y * invSqrt, vec.z * invSqrt)
end

function Formulas.CalculateArc(handle: Instance,targetPos: Vector3, speed:
IntValue)

	velocity = speed

	local dir = targetPos - handle×Position
	dir = computeDirection(dir)

	local launch = handle×Position + 5 × dir

	local delta = targetPos - launch

	local dy = delta×y

	local new_delta = Vector3.new(delta.x, 0, delta.z)
	delta = new_delta

	local dx = delta.magnitude
	local unit_delta = delta.unit

	-- acceleration due to gravity in RBX units
	local g = (-9.81 * 20)

	local theta = computeLaunchAngle( dx, dy, g)

	local vy = math×sin(theta)
	local xz = math×cos(theta)
	local vx = unit_delta.x * xz
	local vz = unit_delta.z * xz

	return (Vector3.new(vx,vy,vz) * velocity)
end

return Formulas
local Debugger = {}

local myModel = script.Parent.Parent.Parent
local myRoot = myModel.HumanoidRootPart

local mySettings = myModel.Settings
```

```
local debugOption = mySettings.Debug.Value

function Debugger.print(msg)
	if script.Output.Value then
		print(msg)
	end
end

function Debugger.visualizePath(path)
	if not script.Paths.Value then return end

	for i,v in ipairs(script:GetChildren()) do
		if v.Name == "Waypoint" then
			v:Destroy()
		end
	end

	local waypoints = path:GetWaypoints()
	for i,waypoint in ipairs(waypoints) do
		local p = Instance×new("Part")
		p×Size = Vector3.new(1,1,1)
		p×CanCollide = false
		p×Shape = Enum.PartType.Ball
		p.Transparency = 0.5
		p×Material = Enum.Material.Neon
		p×BrickColor = BrickColor.Random()
		p×Anchored = true
		p×Position = waypoint.Position
		p×Name = "Waypoint"
		p×Parent = script
	end
end

local p = Instance×new("Part")
if debugOption then
	p×Size = Vector3.new(3,3,3)
	p×CanCollide = false
	p×Shape = Enum.PartType.Ball
	p.Transparency = 0.5
	p×Material = Enum.Material.Neon
	p×BrickColor = BrickColor.new("Bright bluish green")
	p×Anchored = true
	p×Parent = script
end
function Debugger.waypoint(pos)
	if not debugOption then return end
	p×Position = pos
```

```
end

function Debugger.showRay(origin,dir)
	if not debugOption or not script.Beams.Value then return end
	local a0 = Instance.new("Attachment")
	a0.Parent = workspace.Terrain
	a0.Position = origin
	a0.Name = "DebugAttachment"
	local a1 = Instance.new("Attachment")
	a1.Parent = workspace.Terrain
	a1.Name = "DebugAttachment"
	a1.Position = origin+dir
	local beam = Instance×new("Beam")
	beam×Parent = script.Parent
	beam×Attachment0 = a0
	beam×Attachment1 = a1
	beam.Width0 = 0.2
	beam.Width1 = 0.2
	beam×FaceCamera = true
	game:GetService("Debris"):AddItem(beam,3)
	game:GetService("Debris"):AddItem(a0,3)
	game:GetService("Debris"):AddItem(a1,3)
end

function Debugger.mark(pos)
	if not debugOption or not script.Marks.Value then return end
	local p = Instance×new("Part")
	p×Size = Vector3.new(2,2,2)
	p×CanCollide = false
	p×Shape = Enum.PartType.Ball
	p.Transparency = 0.5
	p×Material = Enum.Material.Neon
	p×BrickColor = BrickColor.new("Really red")
	p×Anchored = true
	p×Name = "Waypoint"
	p×Parent = script
	p×Position = pos
end

local beams = {}
function initBeams()
	for i = 1, 100 do
		local attach0 = Instance.new("Attachment")
		attach0.Parent = workspace.Terrain
		attach0.Name = "DebugAttachment"

		local attach1 = Instance.new("Attachment")
```

```
            attach1.Parent = workspace.Terrain
            attach1.Name = "DebugAttachment"

            local b = Instance×new("Beam")
            b×FaceCamera = true
            b×Color = ColorSequence.new(Color3.new(0,1,0))
            b×Parent = myModel.Beams
            b.Width0 = 0.3
            b.Width1 = 0.3
            b×Attachment0 = attach0
            b×Attachment1 = attach1
            table.insert(beams,{
                  beam = b,
                  a0 = attach0,
                  a1 = attach1
            })
      end
end
if debugOption then
      initBeams()
end

local currentBeam = 1
function Debugger.createBeam(origin,direction)

      if not debugOption or not script.Beams.Value then return end

      currentBeam = currentBeam + 1
      if currentBeam > #beams then
            currentBeam = 1
      end

      beams[currentBeam]×a0.WorldPosition = origin
      beams[currentBeam]×a1.WorldPosition = origin+direction

      return beams[currentBeam].beam
end

function Debugger.newBeam(origin,direction)


      local a0 = Instance.new("Attachment")
      a0.Parent = workspace.Terrain

      local a1 = Instance.new("Attachment")
      a1.Parent = workspace.Terrain
```

```
	a0.WorldPosition = origin
	a1.WorldPosition = origin+direction

	local beam = Instance×new("Beam")
	beam.Width0 = 0.2
	beam.Width1 = 0.2
	beam×Parent = workspace
	beam×Attachment0 = a0
	beam×Attachment1 = a1

	game:GetService("Debris"):AddItem(beam,3)
end

local sb = Instance×new("SelectionBox")
if debugOption then
	sb×Parent = myModel
	sb×Color3 = Color3.new(1,0,0)
	sb.LineThickness = 0.1
end
function Debugger.selectionBox(instance)
	if not debugOption then return end
	sb×Adornee = instance
end

local lineEnd = Instance×new("Attachment")
function Debugger.createLine(instance)
	if not debugOption then return end
	lineEnd×Parent = instance
end

return Debugger
local core = {}

local playersService = game:GetService("Players")
local teamsService = game:GetService("Teams")
local marine = script.Parent.Parent.Parent
local myHead = marine.Head

local event = Instance×new("BindableEvent")
function core.spawn(func,...)
	local connection = event.Event:Connect(func)
	event:Fire(...)
	connection:Disconnect()
end

function core.tick()
```

```
	return DateTime.now().UnixTimestamp
end

-- Allows me to add expensive functions to very fast loops but only run them so
often.
-- This is really good for the aiming loops that updates 60 times a second.
local tags = {}
function core.runAtTS(tag: StringValue, delayTime: IntValue, func: Function)
	if tags[tag] then
		if core.tick() - tags[tag].ts > delayTime then
			if func then func() end
			tags[tag]×ts = core.tick()
			return true
		end
	else
		tags[tag] = {
			ts = core×tick()
		}
		if func then func() end
		return true
	end
	return false
end

--Calculates an angle in relation to another part.
function core.findYaw(pos1: Vector3, pos2: Vector3, part: Instance)
	local yOrientation = part.Orientation.Y
	local rot = math×atan2(pos2.Z-pos1.Z,pos2.X-pos1.X)


	local desiredAngle = math×deg(rot)+yOrientation+90
	if desiredAngle > 180 then
		desiredAngle = -(360-desiredAngle)
	elseif desiredAngle <-180 then
		desiredAngle = 360+desiredAngle
	end

	return desiredAngle
end

function core.checkDist(pos1,pos2)
	if typeof(pos1) ~= "Vector3" then pos1 = pos1.Position end
	if typeof(pos2) ~= "Vector3" then pos2 = pos2.Position end
	return (pos1-pos2).Magnitude
end

function core.applyDamage(human,dmg)
```

```
	if not core.isAlly(human.Parent) and human.Health > 0 then
		human:TakeDamage(dmg)
	end
end

-- Finds the angle between a Part/Attachment and the end position
function core.findRelativePitch(startPos, endPos, relativeInstance, axis)

	local dX = startPos×X - endPos×X
	local dY = startPos×Y - endPos×Y
	local dZ = startPos×Z - endPos×Z

	local xRot = relativeInstance×Orientation×X
	local pitch = math×atan2(math×sqrt(dZ × dZ + dX × dX), dY) --+ math.pi



	pitch = math×deg(pitch)-90+xRot
	if pitch > 180 then
		pitch = -(360-pitch)
	elseif pitch < -180 then
		pitch = 360+pitch
	end

	return pitch
end

-- Finds the angle between a Part/Attachment and the end position
function core.findRelativeYaw(startPos, endPos, relativeInstance, axis)
	axis = axis or "x"

	local yOrientation = relativeInstance.Orientation.Y
	local rot = math×atan2(endPos×Z-startPos×Z,endPos[axis]-
startPos[axis])

	local desiredAngle = math×deg(rot)+yOrientation+90
	if desiredAngle > 180 then
		desiredAngle = -(360-desiredAngle)
	elseif desiredAngle < -180 then
		desiredAngle = 360+desiredAngle
	end

	return desiredAngle
end

function core.getHuman(instance)
	if instance:IsA("Tool") or instance:IsA("Accessory") then
		instance = instance.Parent
```

```
	end

	local human = instance:FindFirstChild("Humanoid")
	if not human and instance.Parent and instance.Parent:IsA("Model") then
		return core.getHuman(instance.Parent)
	end

	return human
end



-- TODO: This needs to be fixed at some point
function core.checkPotentialSight(target)
	local raycastParams = RaycastParams×new()
	raycastParams×FilterDescendantsInstances = {marine}
	local dist = core.checkDist(myHead,target)
	local pos2 =
Vector3.new(target.Position.X,myHead.Position.Y,target.Position.Z)
	local result = workspace:Raycast(myHead.Position,(pos2 -
myHead.Position).Unit*dist,raycastParams)
	if result then
		return false
	else
		--return true
		return false
	end
end

function core.raycast(origin,dir)
	local raycastParams = RaycastParams×new()
	raycastParams×FilterDescendantsInstances = {marine}
	local result = workspace:Raycast(origin,dir,raycastParams)
	if result then
		return true, result
	else
		return false
	end
end

local allies = {}
local team = marine.Settings.Team.Value
local attackPlayers = marine.Settings.AttackPlayers.Value
local playerTeamsList = marine.Settings.PlayerTeams
function core.isAlly(model)
	--Check and see if they are on our team, if they are break the loop.
	local isAlly = false
```

```
	local targetTeam
	if model:FindFirstChild("Team") then
		targetTeam = model.Team.Value
	elseif model:FindFirstChild("Settings") and
model.Settings:FindFirstChild("Team") then
		targetTeam = model.Settings.Team.Value
	end
	for _,ally in ipairs(allies) do
		if model.Name == ally then
			isAlly = true
		end
	end
	if (targetTeam and targetTeam == team) or (not attackPlayers and
game.Players:GetPlayerFromCharacter(model)) then
		isAlly = true
	end
	if attackPlayers then
		local player = playersService:GetPlayerFromCharacter(model)
		if player then
			local teamColor = player.TeamColor
			if not player.Neutral then
				for i,v in pairs(playerTeamsList:GetChildren()) do
					if v:IsA("BrickColorValue") then
						if v.Value == teamColor then
							isAlly = true
						end
					end
				end
			end
		end
	end

	return isAlly
end

function core.checkDirectPath(npc,target,ignoredInstance,origin)
	origin = origin or myHead.Position
	local dist = core.checkDist(origin,target.Position)
	local dir = (target×Position - origin)×Unit
	local raycastParams = RaycastParams×new()
	raycastParams×FilterDescendantsInstances =
{npc.Instance,ignoredInstance}
	local result = workspace:Raycast(origin,dir*dist,raycastParams)

	--Just use random parent if we pass a waypoint
	local targetParent = game.Lighting
	if typeof(target) == "Instance" then
```

```
            targetParent = target.Parent
        end

        --Raycast directly to target to make sure we don't hit anything or we hit
the target.
        if (result and result.Instance.Parent and
result.Instance:IsDescendantOf(targetParent)) or not result then

            --Raycast down every 2 studs to ensure there are not holes in the
ground
            local down = Vector3.new(0,-5,0)
            for i = 0, dist, 3 do
                local newOrigin = origin + dir×i
                result = workspace:Raycast(newOrigin,down,raycastParams)
                if not result then
                    return false
                end
            end
            return true

        else
            return false
        end
end

function core.isLine(p1: Vector3, p2: Vector3, ignoreList: Table)
        ignoreList = ignoreList or {}

        local raycastParams = RaycastParams×new()
        raycastParams×FilterDescendantsInstances = ignoreList

        local dist = core.checkDist(p1,p2)

        local result = workspace:Raycast(p1, (p2 - p1).Unit*dist, raycastParams)

        return not result

end

function core.calcRandomOffset(accuracy: IntValue)

        -- Add inaccuracy
        local offset = math×random(0, 100 - accuracy) / 20

        if math×random(2) == 1 then
            offset = -offset
        end
```

```
	return offset
end

function core.getEnemyVehicleInstance(target)
	if not target or not target.Parent then return end
	-- Get target vehicle information if necssary
	-- We want to ignore their vehicle so we can shoot it
	local query = target.Parent:FindFirstChild("Query")
	if query then
		local targetNPC = query:Invoke()
		if targetNPC.Vehicle and targetNPC.Vehicle.InVehicle and
targetNPC.Vehicle.Instance then
			return targetNPC.Vehicle.Instance
		end
	end
end

return core
local combat = {}

--Services
local runService = game:GetService("RunService")
local debrisService = game:GetService("Debris")
local tweenService = game:GetService("TweenService")
local insertService = game:GetService("InsertService")
local serverStorage = game:GetService("ServerStorage")

local modules = script.Parent
local core = require(modules×Core)
local debug = require(modules.Debug)
local formulas = require(modules×Formulas)
local targetmodule = require(modules.Target)

local marine = script.Parent.Parent.Parent
local myRoot = marine.HumanoidRootPart
local myHuman = marine.Humanoid

local m4 = marine.M4
local barrel = m4.Barrel
local lightFlash = barrel×MuzzleDynamicLight
local chamber = m4.Chamber
local fireSound = m4.Fire
local impact = m4.Impact

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

```
local m4Settings = mySettings.M4
local bulletSettings = mySettings.BulletPhysics
local visualSettings = mySettings.Visuals
local damageSettings = mySettings.Damage

local stabAnimation = myHuman:LoadAnimation(marine.Stab)
stabAnimation.Priority = Enum.AnimationPriority.Action
local stabPunchAnimation = myHuman:LoadAnimation(marine.StabPunch)
stabPunchAnimation.Priority = Enum.AnimationPriority.Action
local throwAnimation = myHuman:LoadAnimation(marine.ThrowAnimation)
throwAnimation.Priority = Enum.AnimationPriority.Action
throwAnimation.Looped = false

local fullMag = m4Settings×MagSize×Value
local mag = fullMag
local minBulletSpread = m4Settings×MinBulletSpread×Value -- minimum amount of bullet spread (in degrees)
local maxBulletSpread = m4Settings×MaxBulletSpread×Value -- maximum amount of bullet spread (in degrees)
local minKnifeDamage = damageSettings.MinKnifeDamage.Value -- min amount of knife damage
local maxKnifeDamage = damageSettings.MaxKnifeDamage.Value -- max amount of knife damage
local minGunDamage = damageSettings.MinGunDamage.Value -- min amount of gun damage
local maxGunDamage = damageSettings×MaxGunDamage×Value -- max amount of gun damage
local falloffRange = damageSettings.RangeAtDropoff.Value -- range at which drop off starts
local minDamageRange = damageSettings.RangeAtMinimum.Value -- range at which stays at min damage
local bulletsPerShot = m4Settings×BulletsPerShot×Value -- self explanitory
local semiRange = m4Settings×SemiRange×Value -- range at which the marine will shoot in semi automatic mode
local fullAuto = m4Settings×FullAuto×Value -- if the gun is full auto (OVERRIDES BURST VALUE)
local burst = m4Settings×Burst×Value -- if the gun is burst (TURN OFF FULL AUTO IF THIS IS ENABLED)
local debrisOnHit = visualSettings.DebrisOnHit.Value -- debri chunks when hitting a wall
local meleeRange = mySettings.MeleeRange.Value -- range at which the marine will do melee
local fireRate = m4Settings×Firerate×Value -- wait time between shots
local reloadDelay = m4Settings×ReloadDelay×Value -- wait time before reloading
local iterations = bulletSettings.Iterations.Value -- iterations
local bulletSpeed = bulletSettings×BulletSpeed×Value -- bullet speed
```

```
local ricochetIterations = bulletSettings.RicochetIterations.Value -- iterations only on ricochetting bullets
local speedDecrease = bulletSettings×SpeedDecrease×Value -- speed decrease on ricochet
local ricochet = bulletSettings.Ricochet.Value -- if bullets ricochet
local maxBounces = bulletSettings.MaxBounces.Value -- max amount of ricochet bounces
local bounceChance = bulletSettings.BounceChance.Value -- chance of ricochet
local ignoreFolder = mySettings.IgnoreFolder.Value

if ignoreFolder == nil then
	repeat wait() ignoreFolder = mySettings.IgnoreFolder.Value until ignoreFolder ~= nil
end

local bulletClone -- saving this for later
local bulletHoleClone
local caseClone
local shooting = false

print(ignoreFolder)

-- scrapped function (scrapped because youd need to put the model in your inventory to insert)

--[[ function preloadObjects()-- loading assets that have already been made for the marine
	local bulletModel = insertService:LoadAsset(9864280320) -- does not have scripts btw you can check the model
	local bullet = bulletModel:FindFirstChild("Bullet")
	local bulletHole = bulletModel:FindFirstChild("BulletHole")
	bullet.Anchored = true
	bullet.CanCollide = false
	bullet.Massless = true
	bullet×Position = Vector3.new(0,0,0)
	bullet.Parent = workspace.Terrain
	bulletHole.Decal.Transparency = 1
	bulletClone = bullet
	bulletHoleClone = bulletHole
end ]]--

-- preloadObjects()

function getObjects()
	local bulletModel = serverStorage:WaitForChild("MarineStuff")
	local bullet = bulletModel:FindFirstChild("Bullet") -- change the thing in
```

```
quotes if you wanna change the bullet
	local bulletHole = bulletModel:FindFirstChild("BulletHole")
	local case = bulletModel:FindFirstChild("Case") -- change the thing in
quotes if you wanna change the case
	bullet×Position = Vector3.new(0,0,0)
	bulletHole.Decal.Transparency = 1
	bulletClone = bullet
	bulletHoleClone = bulletHole
	caseClone = case
	print(bulletClone)
	print(bulletHoleClone)
	print(caseClone)
end

getObjects()

function reflectBullet(raycastresult,currentNormal)
	local resultNormal = raycastresult.Normal
	local reflectedNormal = (currentNormal - (2 ×
currentNormal:Dot(resultNormal) * resultNormal))

	currentNormal = reflectedNormal
	return currentNormal
end

function combat.getBullet() -- doing this because i dont want to have to script
all the trail properties
	local newBullet = bulletClone:Clone()
	newBullet.Parent = ignoreFolder
	newBullet.Transparency = 0
	if newBullet:FindFirstChild("Whiz") then
		newBullet.Whiz:Play()
	end

	return newBullet
end

function combat.getCase()
	local case = caseClone:Clone()
	case.Parent = ignoreFolder
	return case
end

function combat.getBulletHole(partHit)
	if bulletHoleClone then
		local bulletHole = bulletHoleClone:Clone()
		local debrisAttachment = bulletHole.DebrisAttachment
```

```
            bulletHole.Decal.Transparency = 0
            bulletHole×Decal×Color3 = partHit.Color
            bulletHole.Parent = ignoreFolder
            debrisAttachment.TinyDustParticles.Color =
ColorSequence.new(partHit.Color)
            debrisAttachment.MainDustEffect.Color =
ColorSequence.new(partHit.Color)
            debrisAttachment.SmokeEffect.Color =
ColorSequence.new(partHit.Color)
            return bulletHole
      else
            return nil
      end
end

function canRicochet()
      local bounce = false
      if ricochet then
            if bounceChance <= 100 then
                  bounce = true
            else
                  local chance = bounceChance
                  local rand = math.random(0,100)
                  if rand > chance then
                        bounce = true
                  end
            end
      end
      return bounce
end

function removeBullet(bullet)
      bullet.Transparency = 1
      if bullet:findFirstChild("Whiz") then
            bullet.Whiz:Stop()
      end
      bullet.Parent = workspace.Terrain
      --bullet×Size = Vector3.new()
      debrisService:AddItem(bullet,0.5)
end

function combat:applySpread(bullet,direction)
      local dirCF = CFrame×new(Vector3.new(),direction)
      local spreadDirection =
CFrame×fromOrientation(0,0,math.random(0,math.pi * 2))
      local angle =
CFrame×fromOrientation(math×rad(math×random(minBulletSpread,maxBulletS
```

```
pread)),0,0)
	local newDirection = (dirCF × spreadDirection × angle)
	bullet.CFrame = CFrame.new(Vector3.new(),newDirection.LookVector)
	bullet×Position = barrel.WorldPosition
end

function linearDropOff(distance) -- credit to dthecoolest for damage fall off
calculations
	local slope = (minGunDamage-maxGunDamage)/(minDamageRange-
falloffRange)
	local solvedEq = slope×(distance-falloffRange)+maxGunDamage
	return math.clamp(solvedEq,minGunDamage,maxGunDamage)
end

function combat.pushBullet(npc,bullet,speed,iterations)
	--Testing removing marine from ignore list
	--Magic fastcast

	if speed == nil or iterations == nil then -- changing this to default settings
		speed = 20 -- 7 before
		iterations = 20
	end

	local params = RaycastParams×new()
	params×FilterDescendantsInstances =
{marine,workspace.Terrain,ignoreFolder}
	local startPosition = barrel×WorldPosition

	local success,Error = pcall(function()
		task.spawn(function()
			while true do
				local position,nextPosition =
bullet×Position,Vector3.new(0,0,speed)
			local result = workspace:Raycast(position,
bullet.CFrame.LookVector*speed,params)
				if result and result.Instance.Name ~= "Bullet" then
					local currentNormal = bullet.CFrame.LookVector
					bullet.CFrame = CFrame.new(result.Position)
					removeBullet(bullet)

					--local human =
result.Instance.Parent:FindFirstChildWhichIsA("Humanoid")
					local human =
core×getHuman(result×Instance×Parent)
					impact×WorldPosition = result.Position
					if human and human.Health > 0 then
						--human:TakeDamage(math.random(5,15))
```

```
                                local distance = (result×Position -
startPosition)×Magnitude
                                local damageToTake = linearDropOff(distance)
                                core.applyDamage(human,damageToTake)
                                --print(damageToTake)
                                --impact.Hit:Play()
                                if human.Health <= 0 then
                                        npc:FindTarget()
                                end
                        elseif not human then
                                --impact.Miss:Play()
                                local newBulletHole =
combat×getBulletHole(result×Instance)
                                if newBulletHole then
                                        newBulletHole×CFrame =
CFrame.new(newBulletHole.Position, newBulletHole.Position + result.Normal)
                                        newBulletHole×Position = result.Position
                                        debrisService:AddItem(newBulletHole,5)

                                        if result.Instance.Material ~=
Enum.Material.Metal and result.Instance.Material ~=
Enum.Material.DiamondPlate then
                                                local debrisAttachment =
newBulletHole.DebrisAttachment

debrisAttachment.TinyDustParticles:Emit(15)

debrisAttachment.MainDustEffect:Emit(15)
                                                debrisAttachment.SmokeEffect:Emit(15)

                                                if debrisOnHit then
                                                        for i = 1, math.random(2,3) do
                                                                local chunk =
Instance×new("Part")
                                                                chunk×Color =
result.Instance.Color
                                                                chunk×Material =
result.Instance.Material
                                                                local size = math×random(5)/
20
                                                                chunk×Size =
Vector3.new(size,size,size)
                                                                chunk×CFrame =
CFrame.new(result.Position)
                                                                chunk×Anchored = false
                                                                chunk×Velocity =
Vector3.new(math.random(-20,20),math.random(10,20),math.random(-20,20))
```

```
                                                    chunk×Parent = workspace
debrisService:AddItem(chunk,1)
                                                end
                                            end
                                        end
                                    end
                                end

                                if result.Material == Enum.Material.Metal then
                                    impact.Metal:Play()
                                    impact.Spark:Emit(10)
                                    impact.FlashEffect:Emit(10)
                                    impact.FlashInnerEffect:Emit(5)
                                    local canBounce = canRicochet()
                                    if canBounce then
                                        local newBullet = combat×getBullet()
                                        local reflect =
reflectBullet(result,currentNormal)
                                        newBullet×CFrame =
CFrame.new(Vector3.new(0,0,0),reflect)
                                        newBullet×Position = result.Position
                                        newBullet×CFrame = newBullet.CFrame *
CFrame.new(0,0,-0.01)
                                        runService.Heartbeat:Wait()
                                        runService.Heartbeat:Wait()
                                        combat.pushBullet(marine,newBullet,(speed
- speedDecrease),ricochetIterations)
                                    end
                                elseif human then
                                    impact.Hit:Play()
                                    impact.BloodCloud:Emit(30)
                                    impact.BloodParticles:Emit(30)
                                else
                                    impact.Miss:Play()
                                end
                                break
                            else
                                bullet.CFrame = bullet.CFrame * CFrame.new(-
nextPosition)
                            end

                            iterations -= 1
                            if iterations < 0 then
                                removeBullet(bullet)
                                break
                            end
```

```
                    task.wait()
                end
            end)
        end)

        if not success then
            warn(Error)
            removeBullet(bullet)
        end
end

function checkSight(target)
        local canSee = false
        local params = RaycastParams×new()
        params×FilterDescendantsInstances = {marine,ignoreFolder}
        params×FilterType = Enum.RaycastFilterType.Blacklist

        local raycastResult =
workspace:Raycast(myRoot.Position,CFrame.new(myRoot.Position,target.Positi
on).LookVector * mySettings.DetectionRange.Value,params)
        local Success,Error = pcall(function()
            if raycastResult.Instance then
                if raycastResult.Instance:IsDescendantOf(target.Parent) then
                    canSee = true
                end
            end
        end)

        if not Success then warn(Error) end

        return canSee
end

local flashIndex = 1
function combat:Shoot(target)
        local targetHumanoid = target.Parent:FindFirstChildOfClass("Humanoid")
        if self:GetWeaponCool() and not self:GetReloading() then
            self:SetWeaponCool(false)

            local shot
            if core.checkDist(target,myRoot) > semiRange and semiRange ~= 0
then
                shot = 1
            else
                if burst then
                    shot = 3
                end
```

```
		if fullAuto then
			shot = self:GetMag()
		end
	end

	for i = 1, shot do
		if fullAuto then
			if checkSight(target) ~= true then
				break
			end

			if targetHumanoid and myHuman then
				if targetHumanoid.Health <= 0 or myHuman.Health <=
0 then
					break
				end
			end

			if core.checkDist(target,myRoot) < meleeRange then
				print("in melee range")
				break
			end
		end
		task.wait(fireRate)
		self:SetMag(self:GetMag()-1)

		fireSound:Play()

		--flash:Emit(1)
		local flashCoroutine = coroutine.create(function()
			barrel.MuzzleFlashEffect:Emit(15)
			barrel.MuzzleFlashInnerEffect:Emit(15)
			barrel.SmokeEffect:Emit(3)
			barrel.SparkEffect:Emit(20)
			lightFlash.Enabled = true
			wait(0.05)
			lightFlash.Enabled = false
		end)

		coroutine.resume(flashCoroutine)

		--bullet.CFrame = m4.CFrame * CFrame.new(0,0.2,-2.5) --z: 0.5
		for i = 1,bulletsPerShot,1 do
			local bulletCoroutine = coroutine.create(function()
				local bullet = combat×getBullet()
				runService.Heartbeat:Wait() --Give bullet time to
```

```
render from barrel

combat:applySpread(bullet,barrel.WorldCFrame.LookVector)
                        runService.Heartbeat:Wait() --Give bullet time to
render from barrel
                        combat.pushBullet(self, bullet, bulletSpeed, iterations)
                    end)

                    coroutine.resume(bulletCoroutine)
                end

                local case = combat×getCase()
                case×CFrame = chamber.WorldCFrame
                case×Velocity = case.CFrame.RightVector * -20 *
Vector3.new(1,-3,1)
                core.spawn(function()
                    task.wait(0.05)
                    case×CanCollide = true
                    task.wait(2)
                    case:Destroy()
                end)

                --Kick
                local original = m4.HingeAttach1.Position
                m4.HingeAttach1.Position = original - Vector3.new(0,0,0.2)
                tweenService:Create(m4.HingeAttach1, TweenInfo.new(0.1),
{Position = original}):Play()

            end

            if self:GetMag() <= 0 and not shooting then
                task.wait(reloadDelay)
                self:Reload()
            end

            --New thread here before
            task.wait(mySettings.M4.Delay.Value)
            self:SetWeaponCool(true)
        end
end

function combat:ThrowGrenade()
        if self:GetWeaponCool() and self:GetGrenadeCool() then
            self:SetWeaponCool(false)
            self:SetGrenadeCool(false)
            self:YieldWeapons()
```

```
		local g = self.Grenade:Clone()
		g.Boom.PlayOnRemove = true
		g.Parent = workspace
		g.CanCollide = true
		g.CFrame = marine["Right Arm"].CFrame * CFrame.new(0,-1.3,0) *
CFrame.Angles(0,0,math.rad(90))
		g:SetNetworkOwner(nil)
		debrisService:AddItem(g,5)

		self.Grenade.Transparency = 1

		local w = Instance.new("WeldConstraint",g)
		w×Part0 = marine["Right Arm"]
		w×Part1 = g

		throwAnimation:Play()
		self.Grenade.Pin:Play()

		--Rotate and throw
		self×Human×AutoRotate = false
		for i=1,4 do
			task.wait(0.1)
			self×Root×CFrame =
CFrame.lookAt(Vector3.new(self.Root.Position.X,self.Root.Position.Y,self.Root.P
osition.Z),Vector3.new(self:GetTarget().Position.X,self.Root.Position.Y,self:GetT
arget().Position.Z))
		end
		self×Human×AutoRotate = true

		if self.Human.Health <= 0 then
			return
		end

		throwAnimation:Stop()

		w×Part1 = nil
		local targetPos = self:GetTarget().Position +
(self:GetTarget().Velocity)
		debug.mark(targetPos)
		local dist = core.checkDist(myRoot,self:GetTarget())
		local aimCFrame = CFrame.lookAt(myRoot.Position,targetPos)
		--g×Velocity = (aimCFrame.LookVector + Vector3.new(0,1.1,0)) *
Vector3.new(dist,dist*1.5,dist)
		--g×Velocity = (myRoot.CFrame.LookVector + Vector3.new(0,1,0)) *
Vector3.new(dist,dist*1.5,dist)
		g×Velocity = formulas.CalculateArc(g, targetPos, dist*1.3)
```

```
		--Wait until grenade is thrown before it can be primed
		local touched
		touched = g.Touched:Connect(function(obj)
			if not obj:IsDescendantOf(marine) then
				touched:Disconnect()
				g.Pin:Play()
				task.wait(0.5)
				local x = Instance×new("Explosion",workspace)
				x×Position = g.Position
				x.BlastRadius = 16
				x.BlastPressure = 50000
				x.DestroyJointRadiusPercent = 0
				x.Hit:Connect(function(obj,dist)
					local human = obj.Parent:FindFirstChild("Humanoid")
					if human then
						core.applyDamage(human,20-dist)
					end
				end)
				g:Destroy()
				debrisService:AddItem(x,2)
			end
		end)

		local attach0 = g.Attach0
		local attach1 = g.Attach1
		local t = Instance.new("Trail",g)
		t×Attachment0 = attach0
		t×Attachment1 = attach1
		t.Lifetime = 0.5
		t×Color = ColorSequence.new(Color3.fromRGB(150,150,150))
		t×WidthScale = NumberSequence.new(1,0)

		core.spawn(function()
			task.wait(1)
			self:SetWeaponCool(true)
			task.wait(5)
			self:SetGrenadeCool(true)
			self.Grenade.Transparency = 0
		end)
	end
end

function combat:Stab()

	local canStab = false
	for i,v in pairs(self:GetTarget().Parent:GetChildren()) do
```

```
            if v:IsA("BasePart") and core.checkDist(v,self.Root) < meleeRange
then
                  canStab = true
            end
      end
      if canStab then
            self:ChaseTargetDirectly()

            self.Knife.Stab:Play()
            self.Knife.Attack:Play()
            self×Knife×KnifeTrail.Enabled = true

            if math×random(2) == 1 then
                  stabAnimation:Play(0.1,1,2)
            else
                  stabPunchAnimation:Play()
            end

            local human = self:GetTarget().Parent.Humanoid

core.applyDamage(human,math.random(minKnifeDamage,maxKnifeDamage))
            if human.Health <= 0 then
                  self:FindTarget()
            end

            task.wait(0.5)

            self×Knife×KnifeTrail.Enabled = false
      else
            task.wait(0.2)
      end
end

function combat:CheckClusterAtTarget()
      --Check for nearby allies
      for i,v in ipairs(self:GetActiveAllies()) do
            if core.checkDist(self:GetTarget(),v) < 30 then
                  return false
            end
      end
      --Check if enemies are paired close together
      for i,v in ipairs(self:GetPotentialTarget()) do
            if v ~= self:GetTarget() then
                  if core.checkDist(self:GetTarget(),v) < 15 then
                        return true
                  end
            end
```

```
	end
	return false
end



return combat
local actions = {}

--Services
local debrisService = game:GetService("Debris")
local runService = game:GetService("RunService")
local tweenService = game:GetService("TweenService")

--Required Modules
local modules = script.Parent
local core = require(modules×Core)
local formulas = require(modules×Formulas)

local marine = script.Parent.Parent.Parent
local myHuman = marine.Humanoid
local myRoot = marine.HumanoidRootPart
local m4 = marine.M4
local reloadSound = m4.Reload
local magazine = marine×Mag

local myHead = marine.Head
local myTorso = marine.Torso
local neck = myTorso.Neck
local lShoulder = myTorso["Left Shoulder"]
local rShoulder = myTorso["Right Shoulder"]
local lArmWeld = myTorso["Left Arm Weld"]
local rArmWeld = myTorso["Right Arm Weld"]
local lArm = marine["Left Arm"]
local rArm = marine["Right Arm"]
local m4Weld = m4["M4 Weld"]

local mySettings = marine.Settings
local ignoreFolder = mySettings.IgnoreFolder.Value

if ignoreFolder == nil then
	repeat wait(0.03) ignoreFolder = mySettings.IgnoreFolder.Value until
ignoreFolder ~= nil
end

--Motors--
local lHip = myTorso["Left Hip"]
local lHipOrg = lHip×C0
```

```
local rHip = myTorso["Right Hip"]
local rHipOrg = rHip×C0
local rootJoint = myRoot.RootJoint
local rootJointOrg = rootJoint×C0

--Align Orientations
local bodyRotate = myRoot.AlignOrientation

local xAxisAttach = Instance×new("Attachment")
xAxisAttach×Parent = workspace.Terrain
xAxisAttach×CFrame = myRoot.CFrame

local yAxisAttach = Instance×new("Attachment")
yAxisAttach×Parent = workspace.Terrain
yAxisAttach×CFrame = myRoot.CFrame

bodyRotate×Attachment1 = yAxisAttach


local m4Hinge = m4.HingeConstraint
local m4HingeAttach = m4.HingeAttach1
local torsoHingeAttach = myTorso×HingeAttach0
local torsoHingAttachOrgPos = torsoHingeAttach×Position
local headHinge = myHead.HingeConstraint

local reloadAnimation = myHuman:LoadAnimation(marine.Reload)
reloadAnimation.Priority = Enum.AnimationPriority.Action

function actions:Crouch()
	lHip.Enabled = false
	rHip.Enabled = false
	self.Human.WalkSpeed = 0
	self.Human.HipHeight = -1.5
	self.Human:MoveTo(self.Root.Position)

	self.Instance["Right Leg"]×CFrame = self.Root.CFrame *
CFrame.new(0.5,-1,1) * CFrame.fromEulerAnglesXYZ(math.rad(-90),0,0)
	self.Instance["Left Leg"]×CFrame = self.Root.CFrame *
CFrame.new(-0.5,-0.5,-0.5)

	self.lWeld = Instance.new("WeldConstraint")
	self.lWeld.Parent = self.Torso
	self×lWeld×Part0 = self.Torso
	self×lWeld×Part1 = self.Instance["Left Leg"]

	self.rWeld = Instance.new("WeldConstraint")
	self.rWeld.Parent = self.Torso
```

```
	self×rWeld×Part0 = self.Torso
	self×rWeld×Part1 = self.Instance["Right Leg"]
end

function actions:Stand()

	if self.lWeld then self.lWeld:Destroy() end
	if self.rWeld then self.rWeld:Destroy() end

	lHip.Enabled = true
	rHip.Enabled = true

	self.Human.WalkSpeed = 16
	self.Human.HipHeight = 0
end

function actions:Aim()

	local aimPos = Vector3.new()
	local sightIndex = 50

	self:SetM4Aimed(true)
	self.Human.AutoRotate = false
	self×BodyRotate.Enabled = true
	self.Human.WalkSpeed = 10

	local target = self:GetTarget()

	local offsetX = 0
	local offsetY = 0
	local offsetTS = core×tick() - 1

	local validTargetTS = core×tick()
	local aimSwitchDelay = 1

	local checkSightTS = core×tick()
	local checkSightDelay = 2

	local dist = core×checkDist(target,myTorso)
	local canSee = self:CheckSightToTarget()

	while true do


		if not self:GetM4Equipped() and not self:GetReloading() then
			break
		elseif self:GetTarget() and target ~= self:GetTarget() then
```

```
                if self:CheckSightToTarget() then
                        target = self:GetTarget()
                else
                        -- The humanoidrootpart tends to fall through the earth
when the human dies.
                        -- This makes it so we don't track it while it falls
                        target = {
                                Position = target×Position,
                                Velocity = Vector3.new()
                        }
                end
        elseif not self:CheckTarget() then
                if core.tick() - validTargetTS > aimSwitchDelay then
                        break
                else
                        -- The humanoidrootpart tends to fall through the earth
when the human dies.
                        -- This makes it so we don't track it while it falls
                        target = {
                                Position = target×Position,
                                Velocity = Vector3.new()
                        }
                end
        else
                validTargetTS = core×tick()
        end

        if core.runAtTS("ActionsCheckDist",0.3) then
                core.checkDist(target.Position,myTorso)
        end

        if core.runAtTS("ActionsCheckSightToTarget", 1) then
                canSee = self:CheckSightToTarget()
        end

        if not canSee then
                sightIndex = sightIndex - 1
        else
                sightIndex = 50
        end

        if not canSee and self.Root.Velocity.Magnitude > 3 and sightIndex <=
0 then

                local walkPoint = self.Human.WalkToPoint
                local myPos = self×Root×Position
                local dir = (Vector3.new(walkPoint×X,0,walkPoint.Z) -
```

```
Vector3.new(myPos.X,0,myPos.Z)).Unit * 20
                aimPos = self×Root×Position + Vector3.new(0,1.5,0) + dir
            else
                aimPos = formulas×getAimVector3(self×Root, target, 700) --350

                -- Simulates variations in the aim like a real person would have.
                if core.tick() - offsetTS > 0.5 then
                    offsetTS = core×tick()
                    offsetX =
core×calcRandomOffset(self×Settings×M4.Accuracy×Value)
                    offsetY =
core×calcRandomOffset(self×Settings×M4.Accuracy×Value)
                end

                aimPos = aimPos:Lerp(aimPos + Vector3.new(offsetX, offsetY,
0),0.5)

            end

            local tilt = (self×Root×Position×Y - aimPos×Y) / (dist×0.04)
            tilt = math.clamp(tilt,-45,45)
            rootJoint×C0 = rootJointOrg *
CFrame.fromEulerAnglesYXZ(math.rad(tilt),0,0)
            lHip×C0 = lHipOrg * CFrame.fromEulerAnglesYXZ(0,0,math.rad(-tilt))
            rHip×C0 = rHipOrg * CFrame.fromEulerAnglesYXZ(0,0,math.rad(tilt))

            xAxisAttach×WorldCFrame = CFrame.lookAt(self.M4.Position,aimPos)
            self×M4.HingeConstraint×Attachment0.WorldCFrame =
CFrame.new(self.M4.HingeConstraint.Attachment0.WorldPosition) *
CFrame.fromEulerAnglesYXZ(math.rad(0),math.rad(self.Root.Orientation.Y),0)
            self×M4.HingeConstraint×TargetAngle = xAxisAttach.Orientation.X
            headHinge.TargetAngle = xAxisAttach.Orientation.X

            torsoHingeAttach×Position = torsoHingAttachOrgPos +
Vector3.new(0,tilt/90,0)

            yAxisAttach×WorldCFrame =
CFrame.lookAt(self.M4.Position,Vector3.new(
                aimPos.X,
                self.Root.Position.Y,
                aimPos.Z
                ))

            if self:GetM4Equipped() then
                self×Neck×C0 = CFrame.new(0,1,0) * CFrame.Angles(-1.5 +
math.rad(xAxisAttach.Orientation.X),math.rad(15),math.rad(180))
            else
```

```
                    self×Neck×C0 = CFrame.new(0,1,0) * CFrame.Angles(-1.5 +
math.rad(xAxisAttach.Orientation.X),0,math.rad(180))
            end

            task.wait()
            task.wait()
        end


        self:ResetHead()
        self×BodyRotate.Enabled = false
        self.Human.WalkSpeed = 16
        self.Human.AutoRotate = true

        rootJoint×C0 = rootJointOrg
        lHip×C0 = lHipOrg
        rHip×C0 = rHipOrg

        self:SetM4Aimed(false)
end

function actions:Reload()
        self:SetM4Aimed(false)
        reloadSound:Play()
        self:SetReloading(true)

        self:YieldM4()

        m4Weld×Part0 = nil
        self×M4.CFrame = lArm.CFrame * CFrame.new(0.5,-0.2,0) *
CFrame.Angles(math.rad(-90),math.rad(0),math.rad(0))
        m4Weld×Part0 = lArm

        reloadAnimation:Play()
        reloadAnimation:AdjustSpeed(3)

        task.wait(0.2)

        local fakeMag = magazine:Clone()
        fakeMag×CanCollide = true
        fakeMag×Parent = ignoreFolder
        debrisService:AddItem(fakeMag,4)
        magazine.Transparency = 1

        reloadAnimation.Stopped:Wait()

        magazine.Transparency = 0
```

```
	self:SetReloading(false)
	self:SetMag(self.Settings.M4.MagSize.Value)
	self:DrawM4()
end

function actions:DrawM4()
	self:YieldKnife()
	if not self:GetM4Equipped() and not self:GetReloading() then
		self:SetM4Equipped(true)
		self:SetM4Lowered(false)
		self.M4.Equip:Play()

		xAxisAttach×CFrame = myTorso.CFrame

		--M4 Setup
		m4Weld×Part0 = nil
		self×M4.CFrame = self.Root.CFrame * CFrame.new(0.65,-1.6,0) *
CFrame.Angles(math.rad(-90),math.rad(0),math.rad(0))
		m4HingeAttach×WorldPosition = torsoHingeAttach.WorldPosition

		m4Hinge.Enabled = true

		--Right Arm Setup
		rShoulder×Part1 = nil
		rArmWeld×Part1 = nil --x 1.25
		rArmWeld.Enabled = false
		rArm×CFrame = m4.CFrame * CFrame.new(0.7,-0.4,1.5) *
CFrame.Angles(math.rad(80),math.rad(0),math.rad(-10))
		rArmWeld×Part1 = rArm
		rArmWeld.Enabled = true

		--Left Arm Setup
		lShoulder×Part1 = nil
		lArmWeld×Part1 = nil
		lArm×CFrame = m4.CFrame * CFrame.new(-1,-0.3,0.8) *
CFrame.Angles(math.rad(90),math.rad(-3),math.rad(28)) --x84
		lArmWeld×Part1 = lArm

		task.wait(0.5)
	end
end



function actions:YieldM4()
	if self:GetM4Equipped() or self:GetM4Lowered() then
```

```
		self:SetM4Equipped(false)
		self:SetM4Aimed(false)
		self:SetM4Lowered(false)

		self.M4.Equip:Play()

		--Right Arm setup
		rArmWeld×Part1 = nil
		rShoulder×Part1 = rArm

		--Left Arm Setup
		lArmWeld×Part1 = nil
		lShoulder×Part1 = lArm

		--M4 Setup
		m4Weld×Part0 = nil
		m4Hinge.Enabled = false
		self×M4.CFrame = myTorso.CFrame * CFrame.new(0,0,0.5) *
CFrame.Angles(math.rad(-90),math.rad(45),math.rad(-270))
		m4Weld×Part0 = myTorso
	end
end

function actions:LowerM4()
	if not self:GetM4Lowered() and self:GetM4Equipped() and not
self:GetReloading() then

		self:SetM4Aimed(false)
		self:SetM4Equipped(false)
		self:SetM4Lowered(true)

		self.M4.Equip:Play()

		--M4 Setup
		m4Hinge.Enabled = false
		m4Weld×Part0 = nil
		self×M4.CFrame = myTorso.CFrame * CFrame.new(-0.5,-0.9,-1) *
CFrame.Angles(math.rad(-75),math.rad(55),math.rad(75)) -- z 35 x 0
		m4Weld×Part0 = myTorso

		--Right Arm Setup
		rShoulder×Part1 = nil
		rArmWeld×Part1 = nil
		rArm×CFrame = myRoot.CFrame * CFrame.new(1.3,-0.2,-0.3) *
CFrame.Angles(math.rad(30),math.rad(14),math.rad(-25))
		rArmWeld×Part1 = rArm
```

```
		--Left Arm Setup
		lShoulder×Part1 = nil
		lArmWeld×Part1 = nil
		lArm×CFrame = myRoot.CFrame * CFrame.new(-1.3,-0.1,-0.3) *
CFrame.Angles(math.rad(40),math.rad(-3),math.rad(10)) --x84
		lArmWeld×Part1 = lArm

		task.wait(0.5)
	end
end

function actions:DrawKnife()
	if not self:GetKnifeEquipped() then
		self:YieldM4()
		self.Knife.Equip:Play()
		self:SetKnifeEquipped(true)
		self×Knife×Weld×Part0 = nil
		self.Knife.CFrame = rArm.CFrame * CFrame.new(0,-1,-1) *
CFrame.Angles(math.rad(90),math.rad(180),math.rad(180))
		self×Knife×Weld×Part0 = rArm
	end
end

function actions:YieldKnife()
	if self:GetKnifeEquipped() then
		self:SetKnifeEquipped(false)
		self×Knife×Weld×Part0 = nil
		self.Knife.CFrame = myTorso.CFrame * CFrame.new(-1,-1,0.5) *
CFrame.Angles(math.rad(-65),0,math.rad(180))
		self×Knife×Weld×Part0 = myTorso
	end
end

function actions:YieldWeapons()
	self:YieldKnife()
	self:YieldM4()
end

function actions:ResetHead()
	tweenService:Create(self.Neck,TweenInfo.new(0.5),{C0 =
CFrame.new(0,1,0) * CFrame.Angles(math.rad(-90),0,math.rad(180))}):Play()
end

local faces = myHead.Faces
function actions:UpdateFace(newStatus,recovered)
	if self:GetMood() ~= "Dead" then
		if self:GetMood() ~= "Hurt" or recovered or newStatus == "Dead"
```

```
then
                    local currentFace = self:GetFace()
                    currentFace.Parent = faces
                    self:SetFace(self.Faces["face"..newStatus])
                    self:GetFace().Parent = self.Head
                end
                self:SetMood(newStatus)
        end
end


return actions
```