Tank Game 3D: Shooting System

A Technical Breakdown in Godot Engine

System Overview

The **Tank Game 3D** shooting system consists of:

1. **Tank Controller**: Handles player input and bullet spawning
2. **Bullet System**: Manages bullet movement and collision detection
3. **Explosion Effects**: Visual and audio feedback on impact
4. **Timing System**: Controls fire rate and bullet lifetime

Scene Structure

```
TankGame3D (Node3D)
├── tank (CharacterBody3D)
│   ├── player (MeshInstance3D)
│   ├── StaticBody3D (Gun barrel)
│   ├── bullet_spawn (Marker3D)
│   └── Timer (Fire rate control)
├── enemy (Marker3D)
└── [Dynamically created during gameplay]
    ├── bullet (Area3D)
    │   ├── MeshInstance3D
    │   └── CollisionShape3D
    └── explosion (GPUParticles3D)
        └── AudioStreamPlayer3D
```

Tank Implementation

The tank is a CharacterBody3D with:

- Movement controls (forward/backward, rotation)
- A gun barrel (StaticBody3D)
- A bullet spawn point (Marker3D)
- A timer for controlling fire rate

Tank Script Properties

```
# From tank.gd
extends CharacterBody3D

@export var speed:float = -1
@export var rot_speed = 10.0

# Shooting system properties
@export var bullet_scene:PackedScene
@export var bullet_spawn:Node3D
@export var fire_rate:int = 10

@onready var timer = $Timer
var can_fire:bool = true
```

Shooting Implementation

```
# From tank.gd
func _ready() -> void:
    # Set up fire rate timer
    timer.wait_time = 1.0 / fire_rate
    timer.connect("timeout", _time_out)

func _time_out():
    can_fire = true

func _physics_process(delta: float) -> void:
    # Movement code...

    # Shooting logic
    if can_fire and Input.is_action_pressed("shoot"):
        var bullet = bullet_scene.instantiate()
        get_parent().add_child(bullet)
        can_fire = false
        bullet.global_position = bullet_spawn.global_position
        bullet.global_rotation = global_rotation
        timer.start()
```

Fire Rate Control

The shooting system uses a timer to control fire rate:

1. Fire rate is defined as shots per second (e.g., 10 shots/second)
2. Timer wait time is calculated as `1.0 / fire_rate`
3. When shooting, the timer starts and `can_fire` is set to false
4. When the timer times out, `can_fire` is set back to true
5. This creates a cooldown between shots

Bullet Scene Structure

```
bullet (Area3D)
├── MeshInstance3D (Visual representation)
├── CollisionShape3D (Collision detection)
└── Timer (Optional lifetime control)
```

The bullet is an Area3D node that:

- Moves in a straight line
- Detects collisions with physics bodies
- Spawns explosion effects on impact
- Destroys itself and the target on collision

Bullet Script

```
# From bullet.gd
extends Area3D

@export var speed:float=20
@export var explosion_scene:PackedScene

func _physics_process(delta: float) -> void:
    var v = global_transform.basis.z
    global_position = global_position + v * speed * delta
    look_at(global_position - global_transform.basis.z)
```

Bullet Collision Handling

```
# From bullet.gd
func _on_body_entered(b: Node3D) -> void:
    # Create explosion effect
    var explosion:GPUParticles3D = explosion_scene.instantiate()
    get_parent().add_child(explosion)
    explosion.global_position = global_position
    explosion.emitting = true

    # Match explosion color to hit object
    var color = b.get_node("MeshInstance3D").get_surface_override_material(0).albedo_color
    var m = StandardMaterial3D.new()
    m.albedo_color = color
    explosion.material_override = m

    # Destroy bullet and target
    self.queue_free()
    b.queue_free()
```

Explosion Effect

The explosion is a GPUParticles3D system with:

- 150 particles with 5-second lifetime
- High explosiveness (0.9) for sudden burst
- Particle trails for visual appeal
- Color ramp for fade-out effect
- Scale curve for size reduction
- Sound effect via AudioStreamPlayer3D

Explosion Scene Structure

```
# From explosion.tscn
[node name="explosion" type="GPUParticles3D"]
material_override = SubResource("StandardMaterial3D_7rj46")
amount = 150
lifetime = 5.0
explosiveness = 0.9
randomness = 0.1
trail_enabled = true
process_material = SubResource("ParticleProcessMaterial_q6qou")
draw_pass_1 = SubResource("BoxMesh_q6qou")

[node name="AudioStreamPlayer3D" type="AudioStreamPlayer3D" parent="."]
stream = ExtResource("1_h6mh1")
autoplay = true
```

Particle System Configuration

The explosion uses a ParticleProcessMaterial with:

```
emission_shape = 3  # Box shape
emission_box_extents = Vector3(0.8, 0.8, 0.8)
direction = Vector3(0, 1, 0)  # Upward direction
spread = 80.0  # Wide spread
initial_velocity_min = 2.0
initial_velocity_max = 10.0
scale_curve = SubResource("CurveTexture_mh3qb")  # Size reduction
color_ramp = SubResource("GradientTexture1D_h6mh1")  # Fade out
```

Shooting System Workflow

1. **Input Detection**: Player presses the shoot button
2. **Fire Rate Check**: System verifies cooldown has elapsed
3. **Bullet Creation**: Bullet is instantiated at spawn point
4. **Bullet Movement**: Bullet travels in straight line
5. **Collision Detection**: Bullet detects collision with object
6. **Explosion Creation**: Explosion effect is spawned at impact
7. **Cleanup**: Bullet and target are destroyed

Godot Systems Used

1. **Physics System**:
    - CharacterBody3D for tank
    - Area3D for bullet collision detection
2. **Input System**:
    - Input.is_action_pressed() for shoot detection
    - Input.get_axis() for movement
3. **Transform System**:
    - global_transform.basis.z for direction vectors
    - global_position for positioning
4. **Timer System**:

Godot Systems Used (cont.)

5. **Instancing System**:
    - PackedScene.instantiate() for creating bullets and explosions
    - add_child() for adding to scene tree
6. **Particle System**:
    - GPUParticles3D for explosion effects
    - ParticleProcessMaterial for particle behavior
7. **Audio System**:
    - AudioStreamPlayer3D for spatial sound effects

Key Design Patterns

1. **Object Pooling (Alternative)**:
    - Current implementation creates/destroys bullets
    - Could be optimized with object pooling
2. **Event-Driven Programming**:
    - Uses signals (body_entered) for collision detection
3. **Component-Based Design**:
    - Separates functionality into reusable components
    - Tank, bullet, and explosion are independent scenes

Performance Considerations

1. **Object Creation/Destruction**:
    - Creating and destroying bullets and explosions can be expensive
    - Object pooling could improve performance
2. **Particle System**:
    - 150 particles per explosion might be excessive
    - Trail effects add visual appeal but increase GPU load
3. **Physics Processing**:
    - Bullet movement and collision detection in _physics_process
    - Ensures consistent behavior regardless of frame rate

Code Optimization Opportunities

1. **Object Pooling**:

```
# Pre-create bullets and explosions
var bullet_pool = []
func get_bullet():
    if bullet_pool.size() > 0:
        return bullet_pool.pop_back()
    return bullet_scene.instantiate()
```

2. **Bullet Lifetime**:

```
# Add timeout to bullets to prevent leaks
func _ready():
    $Timer.start(5.0)  # 5-second lifetime
    $Timer.connect("timeout", queue_free)
```

Potential Enhancements

1. **Projectile Variety**:
    - Different bullet types (homing, explosive, etc.)
    - Varying damage amounts
2. **Visual Improvements**:
    - Muzzle flash effects
    - Bullet trails
    - Impact decals
3. **Gameplay Features**:
    - Ammo system
    - Weapon overheating

Integration with Game Systems

The shooting system integrates with:

1. **Player Control System**:
    - Uses same input system as movement
2. **Enemy System**:
    - Detects and destroys enemies on hit
3. **Visual Effects System**:
    - Creates particle effects on impact
4. **Audio System**:
    - Plays sounds on impact

Conclusion

The Tank Game 3D shooting system demonstrates:

- Simple but effective shooting mechanics
- Integration of physics, input, and visual systems
- Event-driven collision handling
- Dynamic object creation and destruction
- Particle effects for visual feedback

This system serves as a solid foundation for more complex game mechanics.

Thank You!

Tank Game 3D: Shooting System - A Technical Breakdown in Godot Engine

For more information:

- Godot Physics Documentation
- Godot Particles Documentation