Tweens in Godot

A Comprehensive Guide

Master smooth animations and transitions in your games

What is a Tween?

A **Tween** (in-betweening) is an animation technique that smoothly interpolates values over time.

Key Benefits:

- Smooth, professional-looking animations
- Simple code-based animations
- No need for AnimationPlayer for basic transitions
- Dynamic runtime animations
- Easy to control and modify

Basic Tween Creation

```
extends Node2D

func _ready():
    # Create a tween
    var tween = create_tween()

    # Animate position over 2 seconds
    tween.tween_property(self, "position", Vector2(400, 300), 2.0)
```

Key Points:

- `create_tween()` creates a new Tween instance
- Tween runs automatically after creation
- Old syntax (Godot 3.x) used Tween nodes - now built-in!

Tween Property Method

```
tween.tween_property(object, property, final_value, duration)
```

Parameters:

- `object`: The object to animate
- `property`: Property path as string (e.g., "position", "modulate:a")
- `final_value`: Target value
- `duration`: Time in seconds

```
# Example: Fade out a sprite
var tween = create_tween()
tween.tween_property($Sprite, "modulate:a", 0.0, 1.5)
```

Multiple Properties

```
func animate_button():
    var tween = create_tween()

    # Chain multiple property animations (sequential)
    tween.tween_property($Button, "scale", Vector2(1.2, 1.2), 0.3)
    tween.tween_property($Button, "scale", Vector2(1.0, 1.0), 0.2)
    tween.tween_property($Button, "rotation", deg_to_rad(360), 0.5)
```

By default, tweens in a chain execute **sequentially**.

Parallel Tweens

```
func parallel_animation():
    var tween = create_tween()

    # All animations run simultaneously
    tween.set_parallel(true)
    tween.tween_property(self, "position", Vector2(400, 300), 1.0)
    tween.tween_property(self, "rotation", deg_to_rad(180), 1.0)
    tween.tween_property(self, "scale", Vector2(2, 2), 1.0)
```

Use `set_parallel(true)` to make animations run at the same time.

Mixing Sequential and Parallel

```
func complex_animation():
    var tween = create_tween()

    # First move (sequential by default)
    tween.tween_property(self, "position:x", 300, 1.0)

    # Then do these in parallel
    tween.set_parallel(true)
    tween.tween_property(self, "position:y", 400, 0.5)
    tween.tween_property(self, "rotation", TAU, 0.5)

    # Back to sequential
    tween.set_parallel(false)
    tween.tween_property(self, "scale", Vector2.ONE, 0.3)
```

Easing and Transitions

Control the **feel** of your animations:

```
func bounce_animation():
    var tween = create_tween()
    tween.tween_property($Sprite, "position:y", 400, 1.0)\
        .set_trans(Tween.TRANS_BOUNCE)\
        .set_ease(Tween.EASE_OUT)
```

Common Transition Types:

- `TRANS_LINEAR` - Constant speed
- `TRANS_QUAD`, `TRANS_CUBIC` - Polynomial curves
- `TRANS_SINE` - Smooth sine wave
- `TRANS_BOUNCE` - Bouncing effect
- `TRANS_ELASTIC` - Elastic spring effect

Easing Types

```
# Start slow, end fast
.set_ease(Tween.EASE_IN)

# Start fast, end slow
.set_ease(Tween.EASE_OUT)

# Start and end slow, fast in middle
.set_ease(Tween.EASE_IN_OUT)

# Opposite of EASE_IN_OUT
.set_ease(Tween.EASE_OUT_IN)
```

Practical Example: Button Hover

```
extends Button

var tween: Tween

func _on_mouse_entered():
    if tween:
        tween.kill()
    tween = create_tween()
    tween.tween_property(self, "scale", Vector2(1.1, 1.1), 0.2)\
        .set_trans(Tween.TRANS_BACK)\
        .set_ease(Tween.EASE_OUT)

func _on_mouse_exited():
    if tween:
        tween.kill()
    tween = create_tween()
    tween.tween_property(self, "scale", Vector2.ONE, 0.2)
```

Tween Callbacks

```
func animate_with_callback():
    var tween = create_tween()

    tween.tween_property($Sprite, "position:x", 500, 2.0)

    # Execute code when tween finishes
    tween.tween_callback(on_animation_complete)

func on_animation_complete():
    print("Animation finished!")
    $Sprite.modulate = Color.GREEN
```

Use `tween_callback()` to trigger functions at specific points.

Tween Intervals (Delays)

```
func delayed_animation():
    var tween = create_tween()

    # Move immediately
    tween.tween_property($Sprite, "position:x", 300, 1.0)

    # Wait 0.5 seconds
    tween.tween_interval(0.5)

    # Then move again
    tween.tween_property($Sprite, "position:x", 600, 1.0)
```

Perfect for creating pauses between animations!

Looping Tweens

```
func pulsing_effect():
    var tween = create_tween()
    tween.set_loops()  # Infinite loop

    tween.tween_property($Sprite, "scale", Vector2(1.2, 1.2), 0.5)
    tween.tween_property($Sprite, "scale", Vector2.ONE, 0.5)

func limited_loop():
    var tween = create_tween()
    tween.set_loops(3)  # Loop 3 times

    tween.tween_property($Sprite, "rotation", TAU, 1.0)
```

Tween Methods

```
func animate_method_example():
    var tween = create_tween()

    # Call a method with interpolated value
    tween.tween_method(set_health_bar, 100.0, 0.0, 2.0)

func set_health_bar(value: float):
    $HealthBar.value = value
    $Label.text = "HP: %d" % value
```

`tween_method()` calls a function repeatedly with interpolated values - great for custom animations!

From Current Value

```
# Default: tween FROM current value TO target
tween.tween_property($Sprite, "position:x", 500, 1.0)

# Tween FROM specified value TO current
tween.tween_property($Sprite, "position:x", 100, 1.0).from(500)

# Tween FROM current TO specified value (same as default)
tween.tween_property($Sprite, "position:x", 500, 1.0).from_current()
```

The `.from()` method lets you specify the starting value.

Relative Animation

```
func move_relative():
    var tween = create_tween()

    # Move 200 pixels from current position
    tween.tween_property($Sprite, "position:x", 200, 1.0)\
        .as_relative()

    # Can call multiple times to keep moving
    await tween.finished
    tween = create_tween()
    tween.tween_property($Sprite, "position:x", 100, 0.5)\
        .as_relative()
```

`.as_relative()` adds the value to the current value.

Controlling Tweens

```
var tween: Tween

func start_animation():
    tween = create_tween()
    tween.tween_property($Sprite, "position:x", 500, 3.0)

func pause_animation():
    if tween:
        tween.pause()

func resume_animation():
    if tween:
        tween.play()

func stop_animation():
    if tween:
        tween.kill()
```

Speed Control

```
func adjustable_speed():
    var tween = create_tween()

    # Run at 2x speed
    tween.set_speed_scale(2.0)
    tween.tween_property($Sprite, "position:x", 500, 2.0)  # Takes 1 second

    # Run at half speed
    tween.set_speed_scale(0.5)
    tween.tween_property($Sprite, "position:y", 300, 2.0)  # Takes 4 seconds
```

Useful for slow-motion or fast-forward effects!

Practical: Health Bar Animation

```
extends TextureProgressBar

func take_damage(amount: int):
    var new_health = max(0, value - amount)

    var tween = create_tween()
    tween.tween_property(self, "value", new_health, 0.5)\
        .set_trans(Tween.TRANS_QUAD)\
        .set_ease(Tween.EASE_OUT)

    # Flash red
    tween.set_parallel(true)
    tween.tween_property(self, "modulate", Color.RED, 0.1)
    tween.tween_property(self, "modulate", Color.WHITE, 0.1).set_delay(0.1)
```

Practical: Screen Shake

```
extends Camera2D

func shake(intensity: float = 10.0, duration: float = 0.5):
    var original_offset = offset

    var tween = create_tween()

    # Shake for duration
    for i in range(10):
        var shake_offset = Vector2(
            randf_range(-intensity, intensity),
            randf_range(-intensity, intensity)
        )
        tween.tween_property(self, "offset", shake_offset, duration / 10)

    # Return to original position
    tween.tween_property(self, "offset", original_offset, 0.1)
```

Practical: UI Popup

```
extends Panel

func _ready():
    modulate.a = 0
    scale = Vector2(0.5, 0.5)

func show_popup():
    var tween = create_tween()
    tween.set_parallel(true)

    # Fade in
    tween.tween_property(self, "modulate:a", 1.0, 0.3)

    # Scale up with bounce
    tween.tween_property(self, "scale", Vector2.ONE, 0.5)\
        .set_trans(Tween.TRANS_BACK)\
        .set_ease(Tween.EASE_OUT)
```

Practical: Floating Pickup

```
extends Area2D

func _ready():
    var tween = create_tween()
    tween.set_loops()

    # Float up and down
    tween.tween_property(self, "position:y", position.y - 20, 1.0)\
        .set_trans(Tween.TRANS_SINE)\
        .set_ease(Tween.EASE_IN_OUT)
    tween.tween_property(self, "position:y", position.y, 1.0)\
        .set_trans(Tween.TRANS_SINE)\
        .set_ease(Tween.EASE_IN_OUT)

    # Rotate continuously
    tween.set_parallel(true)
    tween.tween_property(self, "rotation", TAU, 3.0)
```

Practical: Typewriter Effect

```
extends Label

func typewriter_text(full_text: String, speed: float = 0.05):
    text = ""

    for i in range(full_text.length()):
        var tween = create_tween()
        tween.tween_callback(add_character.bind(full_text[i]))\
            .set_delay(i * speed)

func add_character(character: String):
    text += character
```

Practical: Object Collection

```
extends Node2D

func collect_coin(coin: Node2D):
    var tween = create_tween()
    tween.set_parallel(true)

    # Move to player
    tween.tween_property(coin, "global_position", global_position, 0.5)\
        .set_trans(Tween.TRANS_BACK)\
        .set_ease(Tween.EASE_IN)

    # Shrink
    tween.tween_property(coin, "scale", Vector2.ZERO, 0.5)

    # Callback to remove
    tween.set_parallel(false)
    tween.tween_callback(coin.queue_free)
```

Common Pitfalls

1. Creating tweens in loops without references:

```
# BAD - old tweens get garbage collected
for i in range(10):
    create_tween()  # Lost reference!

# GOOD - store or bind
var tweens = []
for i in range(10):
    tweens.append(create_tween())
```

Common Pitfalls (cont.)

2. Not killing old tweens:

```
var tween: Tween

func animate():
    # Always kill existing tween first
    if tween:
        tween.kill()

    tween = create_tween()
    tween.tween_property(self, "position", target, 1.0)
```

3. Animating deleted objects:

```
# Check if object is valid
if is_instance_valid(target):
    tween.tween_property(target, "position", end_pos, 1.0)
```

Tween vs AnimationPlayer

Use Tweens when:

- Simple runtime animations
- Dynamic values (calculated at runtime)
- Procedural animations
- UI transitions and effects
- Code-driven gameplay animations

Use AnimationPlayer when:

- Complex multi-track animations
- Frame-by-frame animations
- Need timeline editing in editor
- Multiple synchronized properties

Performance Tips

```
# Good: Reuse tween references
var ui_tween: Tween

func animate_ui():
    if ui_tween:
        ui_tween.kill()
    ui_tween = create_tween()
    # ... animation code

# Good: Use set_process_mode to control when tweens run
tween.set_process_mode(Tween.TWEEN_PROCESS_IDLE)  # Default
tween.set_process_mode(Tween.TWEEN_PROCESS_PHYSICS)  # For physics
```

Waiting for Tweens

```
func sequential_actions():
    var tween = create_tween()
    tween.tween_property($Player, "position", target_pos, 1.0)

    # Wait for tween to finish
    await tween.finished

    print("Player reached destination!")
    trigger_next_event()
```

Use `await tween.finished` to wait for completion.

Cheat Sheet: Quick Reference

```
# Create
var tween = create_tween()

# Basic property animation
tween.tween_property(object, "property", value, duration)

# Easing and transitions
.set_trans(Tween.TRANS_SINE).set_ease(Tween.EASE_OUT)

# Parallel execution
tween.set_parallel(true)

# Delays
tween.tween_interval(seconds)

# Callbacks
tween.tween_callback(function)

# Control
tween.pause() / tween.play() / tween.kill()
```

Best Practices

1. **Always store tween references** if you need to control them later
2. **Kill old tweens** before creating new ones on the same properties
3. **Use appropriate easing** - EASE_OUT for UI, EASE_IN_OUT for movement
4. **Chain smartly** - group related animations
5. **Test different transition types** - small changes make big differences
6. **Await tween.finished** for sequential game logic
7. **Consider performance** - don't create hundreds of tweens simultaneously

Advanced: Custom Interpolation

```
func custom_curve_animation():
    var curve = Curve.new()
    curve.add_point(Vector2(0, 0))
    curve.add_point(Vector2(0.5, 1.5))  # Overshoot
    curve.add_point(Vector2(1, 1))

    var tween = create_tween()
    tween.tween_method(apply_custom_curve.bind(curve), 0.0, 1.0, 2.0)

func apply_custom_curve(t: float, curve: Curve):
    var value = curve.sample(t)
    $Sprite.position.y = start_y + (target_y - start_y) * value
```

Resources & Further Learning

Official Documentation:

- Godot Tween Class Reference
- Godot Animation Tutorial

Experimentation is key!

- Try different transition types
- Combine multiple tweens
- Create your own animation library
- Study professional games for inspiration

Remember: Good animation is about timing and easing!

Thank You!

Key Takeaways:

- Tweens make animations easy and code-friendly
- Easing and transitions control the "feel"
- Chain, parallel, and callback for complex animations
- Always manage tween lifecycle (create, use, kill)

Start experimenting with Tweens in your projects today!

Happy game developing! 🎮