Contents

CS 470 Game Development — Homework

Juicy Clicker

Goal: Build a polished clicker game from scratch using Tweens, dynamic nodes (`.new()`), and `await`. This assignment reinforces the core "juice" concepts from Lecture 14 in a fresh, empty project.

Submission: Zip your Godot project folder and upload it to the course portal before the next class.

---

1. Setup

Create the Project

1. Create a **new Godot project** named "Juicy Clicker"
2. Create a `Node2D` root, rename to `Game`
3. Save as `game.tscn`
4. Add a `Timer` node (Autostart: On, Wait Time: 1.0)
5. Add a `Label` node (Text: "Score: 0", Font Size: 48)
6. Attach script `game.gd` to the root `Game` node

---

2. Challenge 1: The Bouncy Spawner *(Required)*

Goal: Spawn a Godot icon every second, but make it *pop* into existence with a bounce animation.

Steps

1. Connect the Timer's `timeout` signal to `game.gd`
2. In the function, create a `Sprite2D` dynamically:

```
var sprite = Sprite2D.new()
sprite.texture = load("res://icon.svg")
```

```
sprite.position = Vector2(randf_range(50, 350), randf_range(100, 500))
add_child(sprite)
```

3. Add the juice — animate scale from (0,0) to (1,1) with a BOUNCE transition:

```
var tween = create_tween()
# TODO: animate scale from Vector2(0,0) to Vector2(1,1)
#       use TRANS_BOUNCE with EASE_OUT
#       duration: 0.5 seconds
```

Concepts Practiced

- `.new()` for dynamic node creation
- `create_tween()` and `tween_property()`
- Transition curves (`TRANS_BOUNCE`, `EASE_OUT`)

---

3. Challenge 2: The Parallel Pop *(Required)*

Goal: When clicked, the sprite shouldn't just vanish — it should have a juicy death animation with multiple properties animating simultaneously.

Steps

1. Detect clicks using math (no Area2D needed):

```
func _input(event):
    if event is InputEventMouseButton and event.pressed:
        for child in get_children():
            if child is Sprite2D:
                # TODO: check if the click hit this sprite using distance_to()
                # TODO: call pop_sprite() if it's a hit
```

2. Create the parallel tween — scale up, fade out, AND spin all at once:

```
func pop_sprite(sprite: Sprite2D):
    var tween = create_tween()
    # TODO: make the tween run all properties in parallel
    # TODO: scale up to (2, 2) over 0.3s with TRANS_SINE
    # TODO: fade modulate:a to 0.0 over 0.3s with TRANS_SINE
    # TODO: spin rotation to TAU * 2 over 0.3s with TRANS_SINE
    # TODO: await the tween, then free the sprite
```

Concepts Practiced

- Math-based collision detection (`distance_to()`)
- Parallel tweens (multiple properties at once)
- `await` for cleanup

---

4. Challenge 3: Floating Combat Text *(Required)*

Goal: Give visual feedback exactly where the player clicked with floating "+1" text.

Steps

1. Track a score variable:

```
var score = 0
```

2. When a sprite is popped, increment score and update the main label:

```
score += 1
$Label.text = "Score: %d" % score
```

3. Create a dynamic Label at the click position and animate it:

```
# TODO: create a Label node dynamically
# TODO: set its text to "+1", set font size to 32, position it at click_position
# TODO: add it as a child

# TODO: tween it: float upward 50px AND fade modulate:a to 0.0 in parallel
#       duration: 0.8 seconds, use TRANS_SINE
# TODO: await the tween, then free the label
```

Concepts Practiced

- Dynamic UI nodes (`Label.new()`)
- Multi-property parallel tweens
- Floating text animations

---

5. Challenge 4: Tweening the Score *(Stretch Goal)*

Goal: Make the main score counter roll up smoothly instead of instantly snapping.

Steps

1. Replace the simple `score` variable with one that has a setter:

```
var display_score: int = 0:
    set(value):
        # TODO: store the new value
        # TODO: update $Label.text using string formatting
```

2. When scoring, tween the variable instead of setting it directly:

```
func add_score(points):
    var tween = create_tween()
    # TODO: tween display_score to its new value over 0.5s
```

Concepts Practiced

- Property setters
- Tweening raw script variables (not just Node properties)

---

6. What This Assignment Covers

By completing this assignment, you will have built a complete game loop from scratch:

Skill	What You Built
`.new()`	Created Sprite2D and Label nodes dynamically
Tweens	Bouncy spawn, parallel death, floating text
Transitions	SINE, BOUNCE for different feels
`await`	Paused execution for cleanup
Math Collision	`distance_to()` for click detection
Setters	Tweened score variable with property setter

You now understand the core tools for game feel.

---

7. Submission Checklist

Before submitting, verify:

- ☐ Sprites spawn with a bounce animation (Challenge 1)
- ☐ Clicking a sprite triggers a parallel death animation (Challenge 2)
- ☐ Score increments and floating "+1" text appears on click (Challenge 3)
- ☐ Project runs without errors
- ☐ Project folder is zipped and uploaded to the course portal