Lesson 2 Quick Reference Guide

Adding Jumping to Your Platformer

Duration: 40 minutes
Godot Version: 4.5
Prerequisites: Completed Lesson 1

🎯 What We're Adding Today

- ✅ Jump with spacebar
- ✅ Understanding velocity and coordinates
- ✅ Experimenting with jump height
- ✅ Adjusting gravity
- ✅ Making jumping feel GOOD!

💬 Key Vocabulary

Coordinate System: In Godot, Y-axis is flipped:

- **UP = Negative Y** (-10, -400, -600)
- **DOWN = Positive Y** (+10, +100, +980)

JUMP_VELOCITY: The force/speed of the jump (negative = upward)

const: A constant value that never changes during the game

var: A variable that can change during the game

is_on_floor(): Function that checks if player is touching the ground

---

📐 Understanding Coordinates

↑ -Y (UP) ← Negative numbers!

|

|

0 -------- +X (RIGHT)

|

|

↓ +Y (DOWN) ← Positive numbers!

← -X (LEFT)

Remember: Going UP uses negative numbers, like going down to basement floors (B1, B2)!

🔧 **Step-by-Step Instructions**

1. Open Your Project (1 min)

1. Open Godot 4.5
2. Open your Year9_Platformer project
3. Open level_1.tscn
4. Click on **Player** node
5. Open the script (click script icon)

---

2. Uncomment the Jump Code (2 min)

Find these lines:

gdscript

```
# --- JUMP CODE (DISABLED FOR LESSON 1) ---
# We'll uncomment this next lesson to add jumping!
# if Input.is_action_just_pressed("ui_accept") and is_on_floor():
#     velocity.y = JUMP_VELOCITY
```

Remove the # symbols from the last 2 lines:

BEFORE:

gdscript

```
# if Input.is_action_just_pressed("ui_accept") and is_on_floor():
#     velocity.y = JUMP_VELOCITY
```

AFTER:

gdscript

```
if Input.is_action_just_pressed("ui_accept") and is_on_floor():
  velocity.y = JUMP_VELOCITY
```

⚠️ **Important:** Keep the indentation (spaces)! Lines must align with other if statements.

Save: Press **Ctrl+S** (Windows) or **Cmd+S** (Mac)

---

3. Test Your Jump! (2 min)

1. Press **F5** to run the game
2. **Test:**
   - Left/Right arrows still work ✓
   - Press **Spacebar** to jump! ✓
   - Try jumping in mid-air (should NOT work)
   - Try holding spacebar (should only jump once)
3. **Stop the game:** Press **F8**

🎉 **Congratulations! Your character can jump!**

🔍 **Understanding the Code**

gdscript

```
if Input.is_action_just_pressed("ui_accept") and is_on_floor():
```

Breaking it down:

Code Part	What It Does
Input.is_action_just_pressed("ui_accept")	Was spacebar JUST pressed this frame?
and	Both conditions must be true
is_on_floor()	Is player touching the ground?
velocity.y = JUMP_VELOCITY	Set upward velocity (-400 = up!)

Why is_action_just_pressed not is_action_pressed?

- just_pressed = detects single press (tap)
- pressed = detects held button (would jump continuously!)

🧪 **Experimentation Activity (10 min)**

Find the PERFECT jump height!

Find this line at the TOP of your script:

gdscript

```
const JUMP_VELOCITY = -400.0
```

Your mission: Test different values and record results!

Experimentation Table

Copy this into your exercise book:

JUMP_VELOCITY	Jump Height	Feels...	Notes
-400	Default	?	Starting value
-200	?	?	
-300	?	?	
-500	?	?	
-600	?	?	
-___	?	?	Your choice

Testing Process

For each value:

1. **Change the number:**

gdscript

```
const JUMP_VELOCITY = -200.0  // Example
```

2. **Save:** Ctrl+S / Cmd+S
3. **Run:** Press F5

4. **Test:**
    - Jump several times
    - Can you clear the platform?
    - Too high? Too low?
    - Write notes: "Feels floaty" or "Barely leaves ground"
5. **Record in your table**
6. **Try the next value!**

Recommended Values to Try

JUMP_VELOCITY	Effect	Good For
-200	Very short hop	Puzzle games
-300	Low jump	Careful platforming
-400	Medium jump	Balanced (default)
-500	High jump	Action games
-600	Very high	Fast-paced games
-800+	Super jump!	Power-up effect

Questions to Consider

- Does your jump feel too "floaty" (hangs in air too long)?
- Does it feel too "heavy" (barely leaves ground)?
- Can you jump high enough to reach higher platforms?
- Would this work for a fast-paced game? A puzzle game?
- How does it compare to Mario? Sonic? Celeste?

Remember: There's NO single "correct" answer! It depends on your game design.

📊 Constants vs Variables

Two types of data storage:

gdscript

```
const JUMP_VELOCITY = -400.0  # CONSTANT - never changes
var score = 0             # VARIABLE - can change
```

Type	When to Use	Example	Naming
const	Value never changes	Jump height, Speed, Max health	UPPERCASE
var	Value can change	Current score, Health, Position	lowercase

Why use const for JUMP_VELOCITY?

- Jump strength doesn't change during gameplay
- Player always jumps the same height
- Makes code clear and prevents accidental changes

🌍 Adjusting Gravity (Optional)

Want to make your character fall faster or slower?

Find this code:

gdscript

```
if not is_on_floor():
  velocity += get_gravity() * delta
```

Modify it to:

gdscript

```
if not is_on_floor():
  velocity += get_gravity() * delta * 1.5  # Multiply by 1.5
```

Try different multipliers:

Multiplier	Effect	Feels Like
× 0.5	Half gravity	Moon gravity (floaty)
× 1.0	Normal gravity	Default (no change)
× 1.5	Stronger gravity	Falls faster

Multiplier	Effect	Feels Like
× 2.0	Double gravity	Very heavy

Test each one and see how it feels!

🌟 Challenge Activities

Challenge 1: Variable Jump Height

Hold spacebar longer = jump higher!

Add this code:

gdscript

```
# At top with other variables
var jump_released = false


# In the jump section, replace with:
if Input.is_action_just_pressed("ui_accept") and is_on_floor():
   velocity.y = JUMP_VELOCITY
   jump_released = false


# Add RIGHT AFTER jump code:
if Input.is_action_just_released("ui_accept") and velocity.y < 0:
   velocity.y = velocity.y * 0.5  # Cut jump short
```

Test:

- Quick tap = short hop
- Hold longer = full jump

Challenge 2: Double Jump

Jump again while in the air!

gdscript

```
# At top with constants
const MAX_JUMPS = 2
var jump_count = 0

# Replace jump code with:
if is_on_floor():
   jump_count = 0  # Reset when landing

if Input.is_action_just_pressed("ui_accept") and jump_count < MAX_JUMPS:
   velocity.y = JUMP_VELOCITY
   jump_count += 1
```

Test: Jump once, then press spacebar again in mid-air!

Challenge 3: Jump Challenge Platforms

1. Duplicate the platform (select Platform, press **Ctrl+D**)
2. Move it to a different height
3. Create a series of platforms at different heights
4. Test: Can you jump between them?
5. Adjust JUMP_VELOCITY if needed

✅ Success Checklist

```

☐ Jump code uncommented (no # *symbols)*

☐ Spacebar makes player jump

☐ Can't jump in mid-air (works correctly)

☐ Tested at least 3 different JUMP_VELOCITY values

☐ Recorded observations in table

☐ Found a jump height that feels good

☐ Saved final script (Ctrl+S)

☐ Understanding of negative Y = up

---

🎮 **Controls**

| **Key** | **Action** |
|---|---|
| ← | Move left |
| → | Move right |
| **Spacebar** | Jump |
| **F5** | Run game |
| **F8** | Stop game |
| **Ctrl+S / Cmd+S** | Save |

---

🐛 **Troubleshooting**

| **Problem** | **Cause** | **Solution** |
|---|---|---|
| Jump goes DOWN not up | Positive number | Make it negative (-400) |
| Can jump infinitely in air | Missing is_on_floor() | Add and is_on_floor() |
| Holding spacebar = continuous jumps | Wrong input check | Use is_action_just_pressed |
```

Problem	Cause	Solution
Jump too weak	Number too low	Use higher negative (-500, -600)
Jump too strong	Number too high	Use lower negative (-300, -200)
Indentation error	Wrong spacing	Match spaces/tabs with other lines

---

💭 Reflection Questions

Answer these in your exercise book:

1. Why is JUMP_VELOCITY negative instead of positive?
2. What's the difference between const and var?
3. What JUMP_VELOCITY value did you choose? Why?
4. How does gravity affect the "feel" of jumping?
5. Name one game with great jumping. What makes it good?

📝 Code Reference

Complete Jump Code Section

gdscript

```
extends CharacterBody2D


# === MOVEMENT CONSTANTS ===
const SPEED = 300.0
const JUMP_VELOCITY = -400.0  # Change this value!


# === MAIN PHYSICS FUNCTION ===
func _physics_process(delta):


  # --- APPLY GRAVITY ---
```

```
if not is_on_floor():
    velocity += get_gravity() * delta

# --- JUMP CODE ---
if Input.is_action_just_pressed("ui_accept") and is_on_floor():
    velocity.y = JUMP_VELOCITY

# --- GET PLAYER INPUT ---
var direction = Input.get_axis("ui_left", "ui_right")

# --- MOVE THE PLAYER ---
if direction:
    velocity.x = direction * SPEED
else:
    velocity.x = move_toward(velocity.x, 0, SPEED)

# --- EXECUTE THE MOVEMENT ---
move_and_slide()
```

🎓 What You Learned

Today you learned about:

- ✅ Coordinate systems in game engines (Y-axis is flipped!)
- ✅ How velocity controls movement
- ✅ The difference between constants and variables
- ✅ Input detection (just_pressed vs pressed)
- ✅ Game feel and iteration
- ✅ Playtesting methodology

📚 Next Lesson Preview

Coming in Lesson 3: Collectible Coins & Scoring!

- 🪙 Create reusable coin scenes
- ⭐ Track and display score
- 🎯 Learn about scene instancing
- 💾 Organize project files
- 🎨 Add UI elements

Think about: What games have satisfying collectibles? What makes collecting things fun?

Well done! Next lesson: Collectible coins and scoring! 🪙

📋 Playtesting Feedback Form

Your Name: _____
Game Creator: _____

Jump Height

☐ Way too low
☐ A bit too low
☐ Perfect height
☐ A bit too high
☐ Way too high

Jump Feel

☐ Too heavy (falls like a rock)
☐ Just right
☐ Too floaty (hangs in air)

Star Rating

Overall fun: ☆ ☆ ☆ ☆ ☆

One suggestion to improve:

End of Lesson 2 Quick Reference Guide