To avoid me explaining things incorrectly, I will be posting the screenshots of every image of code that will be added to the textbook. Some parts are self-explanatory, some parts should be broken down. I will add a comment underneath an image of code that I think needs to be explained, or I don't know how to break that down, OR I just straight up don't know what this section of code does. If an image doesn't have a breakdown request, feel free to comment on it here down below if you still feel it’s important to explain.

This will save time for pretty much everyone, if we get it right the first time. But will mostly be saving Tony editing time, so he doesn't have to make us go back and change finished parts of the written textbook.

Player setup variables (Basic Player Section 1 : YouTube)----------

Online Docs Request Docs Search Help

```
extends KinematicBody2D

var state = "Idle"
var sub_state = "Default"
var next_anim = ""

export (String, "Right", "Left") var start_direction = "Left"

var gravity = 1100

export (float) var run_speed_max = 150
export (float) var run_speed_acceleration = 10
export (float) var run_speed_deceleration = 9

export (float) var sprint_speed_max = 250
export (float) var sprint_speed_acceleration = 15
export (float) var sprint_speed_deceleration = 10

var sprint_stamina_cost = 0.3

onready var speed_max = run_speed_max
onready var acceleration = run_speed_acceleration
onready var deceleration = run_speed_deceleration

var velocity = Vector2()
```

(24, 1)

Image:1

```
var sprint_stamina_cost = 0.3

onready var speed_max = run_speed_max
onready var acceleration = run_speed_acceleration
onready var deceleration = run_speed_deceleration

var acceleration_sensitivity = 1
var deceleration_sensitivity = 1

var velocity = Vector2()

export (float) var jump_force_preset = 200
onready var jump_force = jump_force_preset
export (float) var jump_duration_max = 1
onready var jump_duration = jump_duration_max
export (float) var jump_count_max = 1
onready var jump_count = jump_count_max

export (float) var stamina_max = 100
onready var stamina = stamina_max
export (float) var stamina_regen_amount = 1

var slope_run_angle = 35
var slope_sprint_angle = 45
var slope_slide_angle = 35
var current_slope_angle = 0

var direction = 0
```

Image:2

Part 2 (Basic Player Section 2: YouTube)----------

```
var direction = 0

func setup():
	$PositionCorrectionAnim.play(start_direction)
	set_state("Idle")

func set_state(new_state):
	match new_state:
		"Idle":
			pass

		"Run":
			pass

		"Sprint":
			pass

		"Jump":
			pass

		"Fall":
			pass

	state = new_state
```

Image:1

Explain why its advantageous to use a Match statement here for animation states here:

```
52 func set_state(new_state):
53     match new_state:
54         "Idle":
55             match sub_state:
56                 "Default":
57                     next_anim = "Idle"
58 
59         "Run":
60             match sub_state:
61                 "Default":
62                     speed_max = run_speed_max
63                     acceleration = run_speed_acceleration
64                     deceleration = run_speed_deceleration
65                     deceleration_sensitivity = 1
66 
67                     next_anim = "Run"
68 
69         "Sprint":
70             speed_max = sprint_speed_max
71             acceleration = sprint_speed_acceleration
72             deceleration = sprint_speed_deceleration
73             deceleration_sensitivity = 0.8
74 
75             next_anim = "Sprint"
76 
77         "Jump":
78             pass
79 
80         "Fall":
```

Image:2

```
69         "Sprint":
70             speed_max = sprint_speed_max
71             acceleration = sprint_speed_acceleration
72             deceleration = sprint_speed_deceleration
73             deceleration_sensitivity = 0.8
74 
75             next_anim = "Sprint"
76 
77         "Jump":
78             jump_force = jump_force_preset
79             next_anim = "Jump"
80 
81             $Sound/JumpSound.play()
82 
83         "Fall":
84             next_anim = "Fall"
85 
86     state = new_state
87 
88 func set_sub_state(new_sub_state):
89     sub_state = new_sub_state
90     set_state(state)
91 
92 
93 
94 
95 
96 
                                                    ( 92,  1)
```

Image:3

```
81            $Sound/JumpSound.play()
82
83        "Fall":
84            next_anim = "Fall"
85
86    state = new_state
87
88 func set_sub_state(new_sub_state):
89    sub_state = new_sub_state
90    set_state(state)
91
92 func _physics_process(delta):
93    apply_gravity(delta)
94    control_check()
95
96    velocity = move_and_slide_with_snap(velocity, get_snap_value(), Vector2.UP, slope_stop_check())
97
98    animation_process()
99
100 func apply_gravity(delta):
101    var gravity_mod = 0
102
103    if state == "Fall":
104        gravity_mod = 300
105
106    velocity.y += (gravity + gravity_mod) * delta
107
108
```

error(94,1): The method "control_check" isn't declared in the current class. 1 (106, 50)

Image:4

Explain why we are using move_and_slide_with_snap here:

Slope_stop_check:

```
83        "Fall":
84            next_anim = "Fall"
85
86    state = new_state
87
88 func set_sub_state(new_sub_state):
89    sub_state = new_sub_state
90    set_state(state)
91
92 func _physics_process(delta):
93    apply_gravity(delta)
94    control_check()
95
96    velocity = move_and_slide_with_snap(velocity, get_snap_value(), Vector2.UP, slope_stop_check())
97
98    animation_process()
99
100 func apply_gravity(delta): ...
107
108 func control_check():
109    var move_right = Input.is_action_pressed("Right")
110    var move_left = Input.is_action_pressed("Left")
111    var jump_start = Input.is_action_just_pressed("Jump")
112    var jump = Input.is_action_pressed("Jump")
113    var sprint = Input.is_action_pressed("Sprint")
114
115
116
```

error(96,1): The method "get_snap_value" isn't declared in the current class. (115, 5)

Image:5

```
95
96      velocity = move_and_slide_with_snap(velocity, get_snap_value(), Vector2.UP, slope_stop_check())
97
98      animation_process()
99
100 > func apply_gravity(delta): ...
107
108 v func control_check():
109     var move_right = Input.is_action_pressed("Right")
110     var move_left = Input.is_action_pressed("Left")
111     var jump_start = Input.is_action_just_pressed("Jump")
112     var jump = Input.is_action_pressed("Jump")
113     var sprint = Input.is_action_pressed("Sprint")
114
115 v   if state in ["Hurt", "Dead"]:
116         return
117
118 v   if move_right:
119         direction = 1
120
121 v   if move_left:
122         direction = -1
123
124 v   if move_right or move_left:
125         accelerate()
126
127 v   if not move_right and not move_left:
128         decelerate()
129
```

```
error(96,1): The method "get_snap_value" isn't declared in the current class.        (128, 21)
```

Image:6

```
121 v   if move_left:
122         direction = -1
123
124 v   if move_right or move_left:
125         accelerate()
126
127 v   if not move_right and not move_left:
128         decelerate()
129
130 v func accelerate():
131 v   if abs(velocity.x) < speed_max:
132 v       if velocity.x > 0 and direction.x == -1 or velocity.x < 0 and direction.x == 1:
133             turn()
134
135     var accelerate_mod = 0
136
137 v   match state:
138 v       "Run":
139 v           if check_ground_angle() <= slope_run_angle:
140                 accelerate_mod = 0.8
141
142 v       "Sprint":
143 v           if check_ground_angle() <= slope_run_angle:
144                 accelerate_mod = 1
145 v           if check_ground_angle() <= slope_sprint_angle:
146                 accelerate_mod = 0.7
147
148     velocity.x += (acceleration * (acceleration_sensitivity + accelerate_mod)) * direction
```

```
error(96,1): The method "get_snap_value" isn't declared in the current class.        (148, 91)
```

Image:7

```
137    match state:
138        "Run":
139            if check_slope_angle() <= slope_run_angle:
140                accelerate_mod = 0.8
141
142        "Sprint":
143            if check_slope_angle() <= slope_run_angle:
144                accelerate_mod = 1
145            if check_slope_angle() <= slope_sprint_angle:
146                accelerate_mod = 0.7
147
148    velocity.x += (acceleration * (acceleration_sensitivity + accelerate_mod)) * direction
149
150 func turn():
151    var default_accel_sensitivity = acceleration_sensitivity
152    acceleration_sensitivity = 2
153    yield(get_tree().create_timer(0.2), "timeout")
154    acceleration_sensitivity = default_accel_sensitivity
155
156 func check_slope_angle():
157    var return_angle = 0
158
159    for collision in get_slide_count():
160        return_angle = rad2deg(acos(get_slide_collision(collision).normal.dot(Vector2.UP)))
161
162    return return_angle
163
164
❮ error(96,1): The method "get_snap_value" isn't declared in the current class.                (160, 92)
```

Image:8
Func turn():

Check_slope_angle is very word and math heavy, might need to break this down:
rad2deg - radian to degree conversion within godot

```
119        direction = 1
120
121    if move_left:
122        direction = -1
123
124    if move_right or move_left:
125        accelerate()
126
127    if not move_right and not move_left:
128        decelerate()
129
130 > func accelerate(): ...
149
150 > func turn(): ...
155
156 > func decelerate(): ...
164
165 > func check_slope_angle(): ...
172
173 func get_snap_value():
174    var return_snap = 0
175
176    if is_on_floor():
177        if not state in ["Jump", "Fall"]:
178            return_snap = 16
179
180
181
```

error(96,1): The method "slope_stop_check" isn't declared in the current class. (178, 29)

Image:9

```
112    var jump = input.is_action_pressed("Jump")
113    var sprint = Input.is_action_pressed("Sprint")
114
115    if state in ["Hurt", "Dead"]:
116        return
117
118    if move_right:
119        direction = 1
120
121    if move_left:
122        direction = -1
123
124    if move_right or move_left:
125        accelerate()
126
127    if not move_right and not move_left:
128        decelerate()
129
130 > func accelerate(): ...
149
150 > func turn(): ...
155
156 func decelerate():
157    if velocity.x > 0:
158        velocity.x -= deceleration * deceleration_sensitivity
159    if velocity.x < 0:
160        velocity.x += deceleration * deceleration_sensitivity
161
162
```

error(96,1): The method "get_snap_value" isn't declared in the current class. (162, 5)

Image:10

```
96      velocity = move_and_slide_with_snap(velocity, get_snap_value(), Vector2.UP, slope_stop_check())
97
98      animation_update()
99
100 > func apply_gravity(delta): ...
107
108 > func animation_update(): ...
113
114 > func control_check(): ...
137
138 > func accelerate(): ...
157
158 > func turn(): ...
163
164 > func decelerate(): ...
172
173 > func check_slope_angle(): ...
180
181 v func get_snap_value():
182     var return_snap = 0
183
184     if is_on_floor():
185         if not state in ["Jump", "Fall"]:
186             return_snap = 16
187
188     return Vector2(0, return_snap)
189
190 > func slope_stop_check(): ...
192
```

3 (138, 1)

Image:11

```
189
190 > func slope_stop_check(): ...
192
193 v func state_check():
194     if is_on_floor():
195         if state in ["Jump", "Fall"]:
196             landed()
197
198         if state == "Idle":
199             if velocity.x != 0:
200                 set_state("Run")
201
202         if state in ["Run", "Sprint"]:
203             if velocity.x == 0:
204                 set_state("Idle")
205
206     if direction == 1:
207         $PositionCorrectionAnim.play("Right")
208     elif direction == -1:
209         $PositionCorrectionAnim.play("Left")
210
211 v func landed():
212     jump_count = jump_count_max
213     jump_duration = jump_duration_max
214
215     set_sub_state("Default")
216     set_state("Idle")
217
```

3 (216, 22)

Image:12

```
125        direction = 1
126
127    if move_left:
128        direction = -1
129
130    if move_right or move_left:
131        accelerate()
132
133    if not move_right and not move_left:
134        decelerate()
135
136    if jump_start:
137        jump_start()
138
139    if jump and state == "Jump" and sub_state != "JumpOver":
140        jump_movement()
141
142    if state == "Jump" and sub_state != "JumpOver":
143        if not jump or not can_jump():
144            jump_count -= 1
145            jump_duration = jump_duration_max
146
147            set_sub_state("JumpOver")
148
149    velocity.x = clamp(velocity.x, -speed_max, speed_max)
150
151 > func accelerate(): ...
170
171 > func turn(): ...
```

error(137,1): The method "jump_start" isn't declared in the current class. (140, 24)

Image:13 (adding to previous Jump Code)

```
185
186 > func check_slope_angle(): ...
193
194 > func get_snap_value(): ...
202
203 > func slope_stop_check(): ...
205
206 > func state_check(): ...
223
224 func jump_start():
225     set_sub_state("Default")
226     set_state("Jump")
227     velocity.y = -jump_force
228
229 func jump_movement():
230     velocity.y -= jump_force
231
232     jump_duration -= 0.1
233
234     if jump_duration <= 0:
235         jump_count -= 1
236         jump_duration = jump_duration_max
237
238 > func landed(): ...
244
245
246
247
```

error(143,1): The method "can_jump" isn't declared in the current class. (226, 8)

Image:14

```
203 > func slope_stop_check(): ⊖
205
206 > func state_check(): ⊖
223
224 v func jump_start():
225     set_sub_state("Default")
226     set_state("Jump")
227     velocity.y = -jump_force
228
229 v func jump_movement():
230     velocity.y -= jump_force
231
232     jump_duration -= 0.1
233
234 v   if jump_duration <= 0:
235         jump_count -= 1
236         jump_duration = jump_duration_max
237
238 > func landed(): ⊖
244
245 v func can_jump():
246 v   if jump_count > 0:
247         return true
248
249     return false
250
251
252
```

Image:15

Explanation of func jump_movement:

```
99
100 > func apply_gravity(delta): ⊖
107
108 > func animation_update(): ⊖
113
114 v func control_check():
115     var move_right = Input.is_action_pressed("Right")
116     var move_left = Input.is_action_pressed("Left")
117     var jump_start = Input.is_action_just_pressed("Jump") and can_jump()
118     var jump = Input.is_action_pressed("Jump") and can_jump()
119     var sprint = Input.is_action_pressed("Sprint")
120
121 v   if state in ["Hurt", "Dead"]:
122         return
123
124 v   if move_right:
125         direction = 1
126
127 v   if move_left:
128         direction = -1
129
130 v   if move_right or move_left:
131         accelerate()
132
133 v   if not move_right and not move_left:
134         decelerate()
135
136 v   if jump_start:
137         jump_start()
```

Image:16

Adding jump conditions

```
206 › func slope_stop_check():
208
209
210 func state_check():
211     if is_on_floor():
212         if state in ["Jump", "Fall"]:
213             landed()
214
215         if state == "Idle":
216             if velocity.x != 0:
217                 set_state("Run")
218                 $Sound/RunSound.play()
219
220         if state in ["Run", "Sprint"]:
221             if velocity.x == 0:
222                 set_state("Idle")
223                 $Sound/RunSound.stop()
224
225     if not is_on_floor():
226         $Sound/RunSound.stop()
227
228     if state in ["Hurt", "Dead"]:
229         $Sound/RunSound.stop()
230
231     if direction == 1:
232         $PositionCorrectionAnim.play("Right")
233     elif direction == -1:
234         $PositionCorrectionAnim.play("Left")
235
```

Image:17
Adding sounds to our game

Image:18 Input Settings

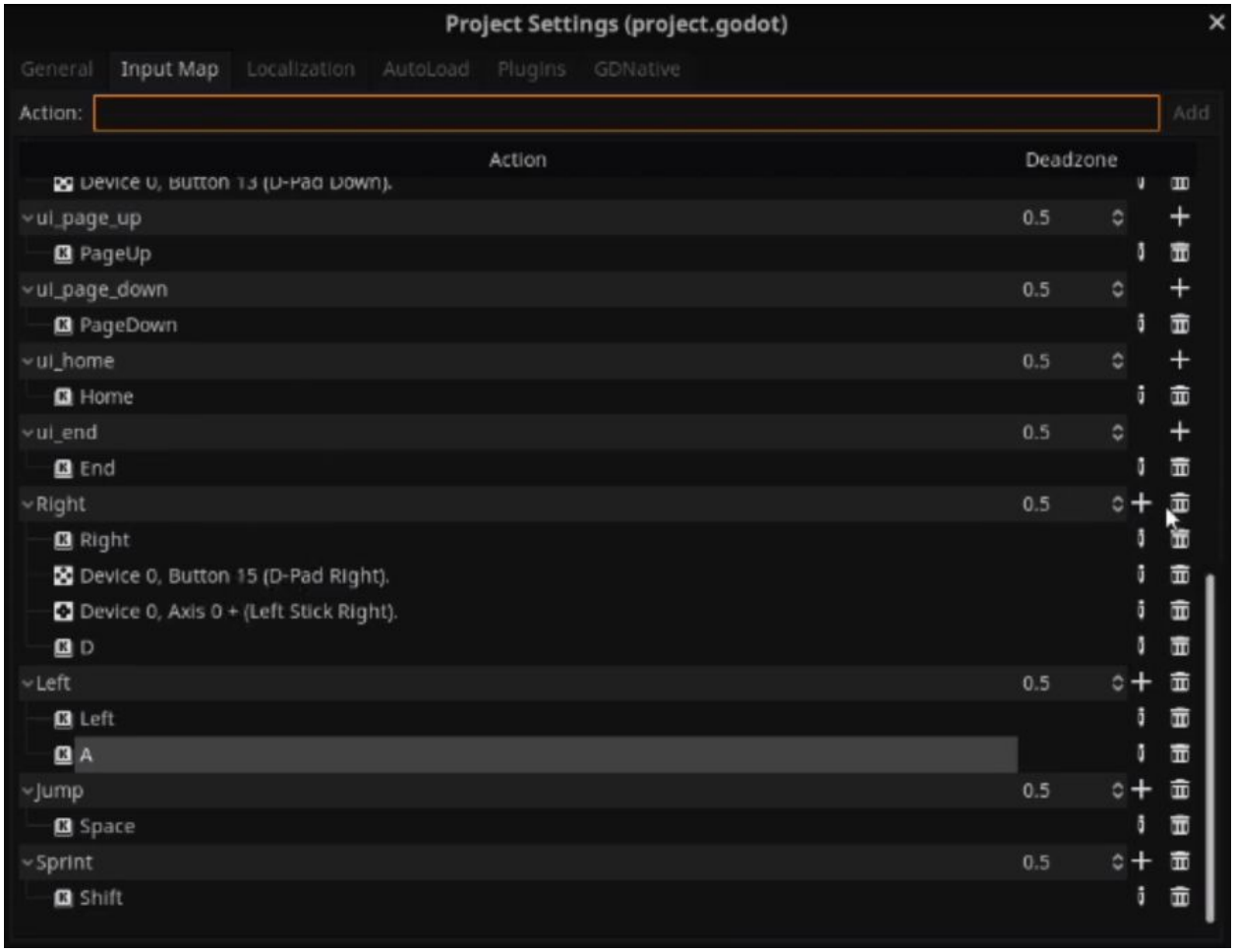

Image:19 adding various input choices

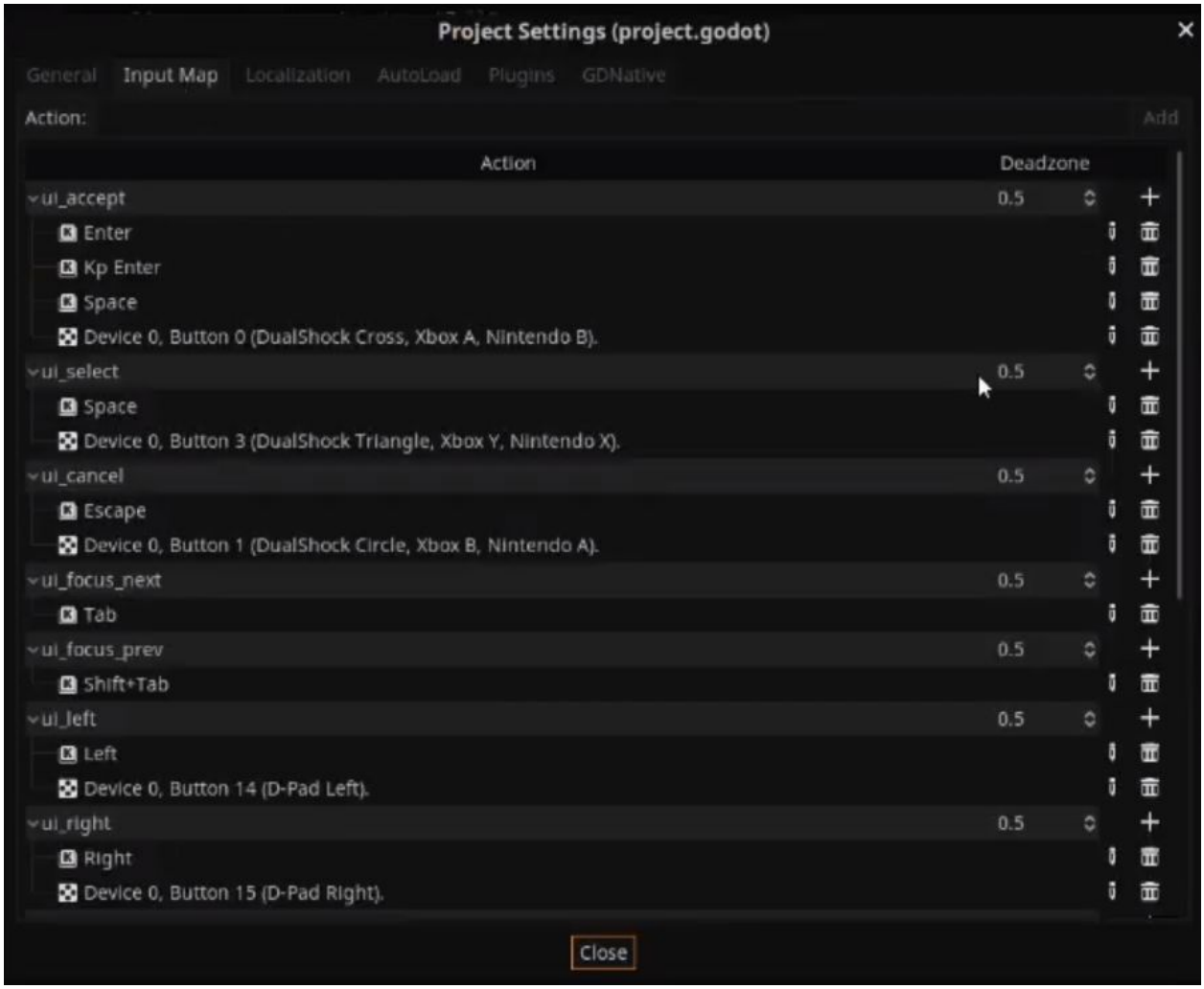

Image:20 More Input Settings

If I missed any code, Let me know, and I'll get those screenshots added. All these photos can be accessed in the Textbook_images folder. They are labeled “Player_Code(#)” [and InputSettings] based on the chronological order they will appear in the book.