Factory Game (221508)

In this 3D exploration game, the player navigates through an ~abandoned~ industrial robotics factory to find and fix 10 scattered robots before the timer runs out.

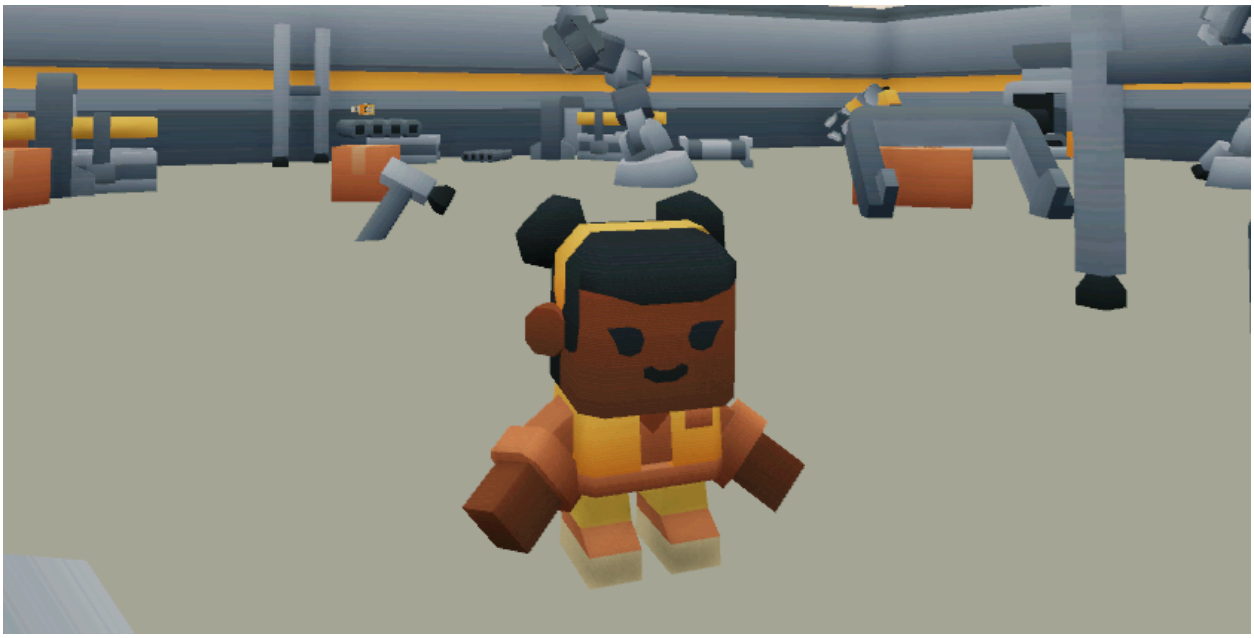
The player can:

- walk with WASD & jump with SPACE;
- interact with the robots by walking over them;
- track how many robots they have saved by the score shown in the top left corner;
- track how much time they have left by the timer label in the top right corner.

The game can be won, or lost depending on if the player manages to collect and "fix" all 10 robots.

Movement

The player is a CharacterBody3D node, and I have used assets from Kenney regarding how the player looks as well as the other static models.



The player moves using
WASD or arrow keys, with gradual acceleration).

```
@export var speed = 5.0
@export var acceleration = 4.0
@export var jump_speed = 20.0
@export var rotation_speed = 12.0
@export var mouse_sensitivity = 0.005
```

Movement direction is aligned with the camera by rotating the input vector based

on the camera’s angle.

```
func _unhandled_input(event):
    if event is InputEventMouseMotion:
        spring_arm.rotation.x -= event.relative.y * mouse_sensitivity
        spring_arm.rotation_degrees.x = clamp(spring_arm.rotation_degrees.x, -
40, 60.0)
        spring_arm.rotation.y -= event.relative.x * mouse_sensitivity
    if event is InputEventKey and event.pressed and event.keycode == KEY_ES
CAPE:
        Input.set_mouse_mode(Input.MOUSE_MODE_VISIBLE)
```

Note: The user can also see their mouse by pressing the ESC key,

Jumping is allowed only when on the ground, applying an upward velocity defined by jump_speed.

```
func _physics_process(delta):
    if not is_on_floor():
        velocity.y = -gravity * delta * 0.5
    else:
        velocity.y = -gravity *delta * 2.0

    get_movement_input(delta)

    if Input.is_action_just_pressed("jump") and is_on_floor():
        velocity.y = jump_speed
        animation_tree.set("parameters/StateMachine/current", "jump")

    move_and_slide()

    if velocity.length() > 1.0:
        model.rotation.y = lerp_angle(model.rotation.y, spring_arm.rotation.y,
```

```
        rotation_speed * delta)
```

Gravity is applied differently when the player is airborne or grounded to give a more natural jump/fall feel.

For the general movement, I have added the WASD keys in the input map, with the names below.

By checking what key the user presses, the character turns that direction to match the camera, and slowly slides using lerp(). Animations have been done using the BlendSpace2D tool for smooth transitions, so we call the animation tree and give it the speed.

```
func get_movement_input(delta):
    var v = velocity.y
    velocity.y = 0
    var input = Input.get_vector("move_left", "move_right", "move_forward", "m
ove_back")
    var direction = Vector3(input.x, 0, input.y).rotated(Vector3.UP, spring_arm.r
otation.y)
    velocity = lerp(velocity, direction * speed, acceleration * delta)
    var vl = velocity * model.transform.basis
    animation_tree.set("parameters/IW/blend_position", Vector2(vl.x, -vl.z) / sp
eed)
    velocity.y = v
```

After, we put the gravity back.

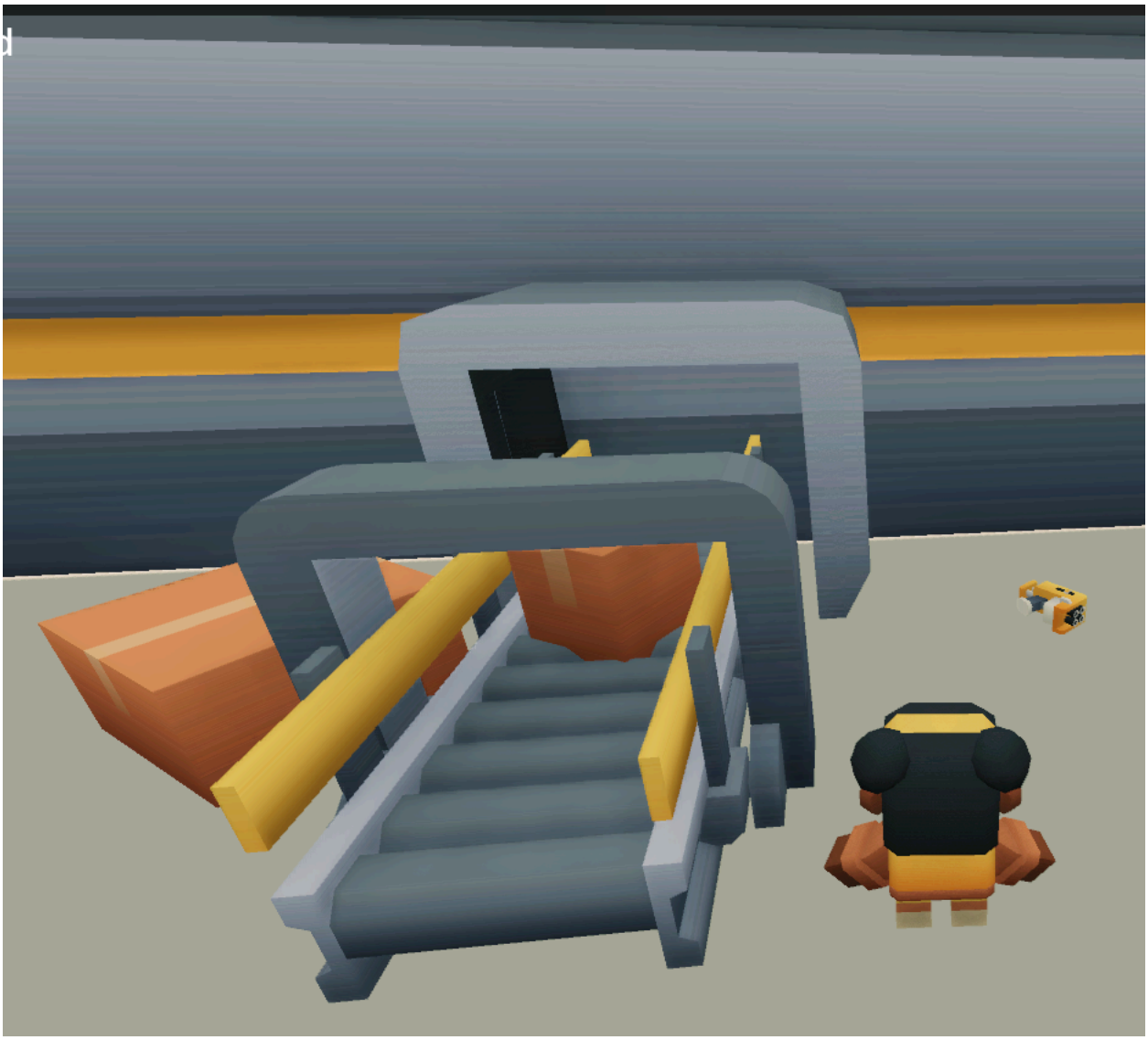
Collecting Robots

To keep track of how many robots we have to save, and how many we have saved I used:

```
var rescued := 0const TOTAL_ROBOTS := 10
```

When the player collides with a robot object, the robot.gd script calls the function collect_robots() from the player.gd script like so:

```
func _on_body_entered(body: Node3D) -> void:
    body.call("collect_robots")
    call_deferred("queue_free")
```



The collect_robots() function increments the amount of rescued robots, updating the label for the user to track their progress.

```
func collect_robots():
    rescued += 1
    saved_label.text = " %d/10 rescued" % rescued
    if (rescued == TOTAL_ROBOTS):
```

```
        Input.set_mouse_mode(Input.MOUSE_MODE_VISIBLE)get_tree()
        .change_scene_to_file("res://game_won.tscn")
```

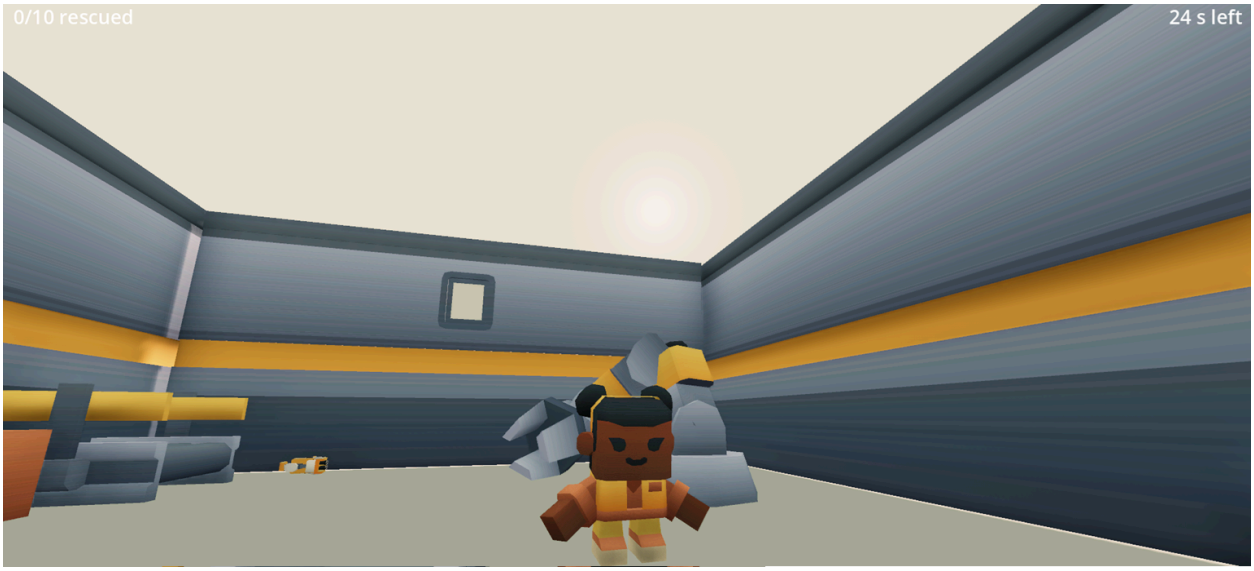
After each collected robot, we check if the user has collected all of them, and if so, we redirect the user to the winning screen.

Timer

On every timer tick in the interval of a second, the function _on_timer_tiimeout() is called, like so:

```
func _on_timer_timeout() -> void:
    time_left -= 1
    timer_label.text = "%d s left " % time_left
    if time_left <= 0:
        if rescued < TOTAL_ROBOTS:
            Input.set_mouse_mode(Input.MOUSE_MODE_VISIBLE)
            get_tree().change_scene_to_file("res://game_over_screen.tscn")
```

We decrement the seconds left, and we update the label so the user can track how much time they have left. If the user runs out of time before collecting all the robots, they are redirected to the loss scene.



Credits

- Models/Assets from: Kenney
- Music from: HitsLab