

GameMaker Lesson: Endless Runner



High School Game Design

Intermediate

GameMaker / GML

1. Lesson Overview

In this lesson, students will create a 2D Endless Runner using the GameMaker game engine. The player controls a character that runs automatically through a scrolling environment while jumping over obstacles. As the game progresses, obstacles appear more frequently, the game becomes faster, and the player earns points for surviving.

The game ends when the player collides with an obstacle. Students will incorporate a restart feature so the player can attempt to beat their previous score.

Project filename: `PX_lastname_EndlessRunner`

Estimated completion time: 3–5 class periods (45–60 minutes each).

Prerequisites: Students should have basic experience with GameMaker sprites, objects, rooms, events, variables, and simple GML programming.

2. Learning Objectives

By the end of this lesson, students will be able to:

- Create the illusion of continuous movement using scrolling backgrounds.
- Program automatic player movement and jumping.
- Spawn obstacles at randomized intervals.
- Increase game speed and difficulty over time.
- Implement collision detection and a game-over condition.

- Design a scoring system based on survival time.
- Test, debug, and improve gameplay.

3. Game Requirements



Automatic running and jumping

The character appears to run continuously. Press Space to jump over obstacles. The player cannot jump again while airborne.



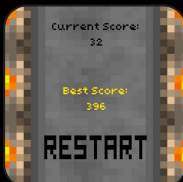
Scrolling environment

The background moves continuously from right to left, creating the appearance of forward movement.



Random obstacles

Obstacles spawn at random intervals and move toward the player. Obstacles that leave the screen are automatically destroyed.



Increasing difficulty and game over

Game speed increases over time. The score is displayed on the screen, and an obstacle collision ends the game with an option to restart.

4. Project Setup

Open GameMaker and create a new project using the GML Code option.

Name your project:

PX_lastname_EndlessRunner

Create a room called `rm_game` with the following settings:

Property	Value
Room width	960 pixels
Room height	540 pixels
Game speed	60 FPS

Ground level	Y = 450
Background	Light blue or a custom background

Create the following resources:

Resource	Name	Purpose
Sprite	<code>spr_player</code>	Player character
Sprite	<code>spr_obstacle</code>	Obstacle or crate
Sprite	<code>spr_background</code>	Scrolling scenery
Object	<code>obj_player</code>	Controls jumping
Object	<code>obj_obstacle</code>	Controls obstacle movement
Object	<code>obj_controller</code>	Controls game speed, scoring, and obstacle spawning

For this lesson, use a 48 × 48 pixel player sprite and a 40 × 40 pixel obstacle sprite. Set the origins of both sprites to Bottom Centre.

Set the background sprite to 960 × 540 pixels, with its origin at Top Left. Design it so the left and right edges connect smoothly when it repeats.

5. Build the Player Character



Create `obj_player` and assign `spr_player`.

The player stays in approximately the same horizontal position throughout the game. The scrolling background and moving obstacles create the illusion of running.

Step 1: Create Event

Add a Create Event to `obj_player` and enter the following code:

JavaScript

```
ground_y = 450;

// Vertical movement
vertical_speed = 0;

// Jumping settings
gravity_force = 0.7;
jump_strength = -13;

// Player begins on the ground
on_ground = true;
```

Step 2: Step Event

Add a Step Event to `obj_player` and enter:

JavaScript

```
// Press Space to jump
if (keyboard_check_pressed(vk_space) && on_ground)
{
    vertical_speed = jump_strength;
    on_ground = false;
}

// Apply gravity
vertical_speed += gravity_force;

// Update vertical position
y += vertical_speed;

// Stop the player at ground level
if (y >= ground_y)
{
    y = ground_y;
    vertical_speed = 0;
    on_ground = true;
}
```

Step 3: Place the player

Open `rm_game` and place one instance of `obj_player` at:

- X = 150
- Y = 450

Run the game. Press Space and verify that the character jumps and returns to the ground.

Checkpoint 1

Show your teacher: Demonstrate that your character can jump, lands correctly, and cannot jump a second time while in the air.

6. Create the Scrolling Background

The background should move from right to left to simulate continuous forward movement.

Open `rm_game` and create a background layer. Assign `spr_background` as the layer's background sprite.

In the Room Editor, enable horizontal tiling for the background layer. Rename the layer exactly:

`Background`

The controller will move this layer using GameMaker's background scrolling functions.

7. Create the Game Controller

Create an object called `obj_controller`. No sprite is required.

Place one instance of `obj_controller` in the room.

Step 1: Create Event

JavaScript

```
// Game settings
game_speed = 5;
max_speed = 15;

speed_increase = 0.002;

// Scoring
score = 0;
```

```
// Obstacle spawning
spawn_timer = 90;

// Game status
game_over = false;

// Scrolling background
background_layer = layer_get_id("Background");
background_element = layer_background_get_id(background_layer);
```

Step 2: Step Event

JavaScript

```
// Restart the game when R is pressed
if (game_over)
{
    if (keyboard_check_pressed(ord("R")))
    {
        room_restart();
    }

    exit;
}

// Increase difficulty gradually
game_speed += speed_increase;

// Limit maximum speed
game_speed = min(game_speed, max_speed);

// Increase score
score += 1;

// Scroll the background
layer_background_xspeed(
    background_element,
    -game_speed * 0.4
);

// Count down until next obstacle
spawn_timer -= 1;

// Spawn a new obstacle
if (spawn_timer <= 0)
{
    var obstacle = instance_create_layer(
        room_width + 40,
```

```

        450,
        "Instances",
        obj_obstacle
    );

    obstacle.move_speed = game_speed;

    // Shorter intervals as the game gets faster
    spawn_timer = irandom_range(80, 130)
        * (5 / game_speed);

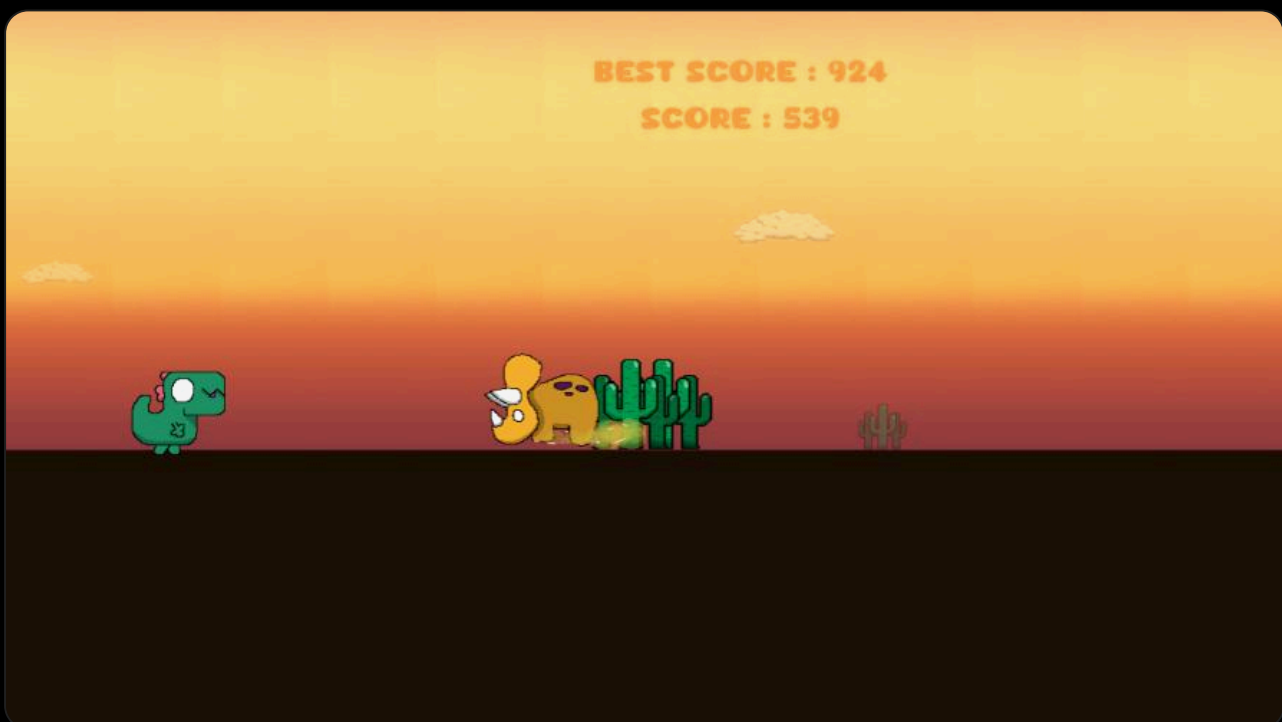
    spawn_timer = max(45, round(spawn_timer));
}

```

Important: The room must contain an instance layer named `Instances`. If your instance layer has another name, either rename it or update the code.

The value `game_speed` is a variable representing the scrolling speed in pixels per game step. It does not change GameMaker's actual frame rate.

8. Program the Obstacles



Create `obj_obstacle` and assign `spr_obstacle`.

Step 1: Create Event

JavaScript

```
move_speed = 5;
```

Step 2: Step Event

JavaScript

```
// Match the current game speed
move_speed = obj_controller.game_speed;

// Move from right to left
x -= move_speed;

// Remove obstacles after they leave the room
if (x < -sprite_width)
{
    instance_destroy();
}
```

The controller creates obstacle instances automatically at randomized intervals. Each obstacle moves toward the player and disappears after leaving the screen.

Step 3: Collision with the player

In `obj_obstacle`, add a Collision Event with `obj_player`.

Enter:

JavaScript

```
obj_controller.game_over = true;
```

When the player touches an obstacle, the controller changes the game's status to Game Over.

The player and obstacles may still overlap visually, but the gameplay stops. Students can add animations, sounds, or other effects later.

Checkpoint 2

Show your teacher: Demonstrate that obstacles appear automatically, travel toward the player, and trigger Game Over when a collision occurs.

9. Create the Score and Game-Over Screen

Open `obj_controller` and add a Draw GUI Event.


```
}  
  
// Restore drawing settings  
draw_set_halign(fa_left);  
draw_set_valign(fa_top);  
draw_set_color(c_white);
```

The Draw GUI Event displays the score independently of the room's camera position.

When the game ends, the score stops increasing and the screen displays instructions for restarting.

10. Test Your Game

Students must test the following conditions before submitting their projects.

Game testing checklist

0 / 10 completed

- ☐ The game launches without error messages.
- ☐ The player jumps when Space is pressed.
- ☐ The player cannot jump again while airborne.
- ☐ The background scrolls continuously.
- ☐ Obstacles appear at different intervals.
- ☐ Obstacles move toward the player and disappear after leaving the screen.
- ☐ The game becomes faster as the player survives.
- ☐ The score increases during gameplay.
- ☐ Colliding with an obstacle produces a Game Over screen.
- ☐ Pressing R after Game Over restarts the game.

11. Additional Challenges

Students who finish the required game may improve their project by implementing one or more of the following features.

Challenge	Description
Double jump	Allow the player to jump twice before touching the ground.
Multiple obstacles	Introduce obstacles with different sizes or appearances.
Collectible coins	Award bonus points for collecting coins while running.

High score	Save and display the highest score achieved during the current game session.
Sound effects	Add sounds for jumping, scoring, and collisions.
Animated player	Create running and jumping animations.
Moving obstacles	Introduce obstacles that move vertically or follow different paths.
Start menu	Add a title screen with instructions and a Start button.

12. Submission Requirements

Submit the following items through Google Classroom.

1. GameMaker project

- Submit the completed project named `PX_lastname_EndlessRunner`, including its necessary project resources.

2. Screenshot

- Submit a screenshot showing your game running inside the GameMaker engine, with the player, scrolling environment, obstacles, and score visible.

3. Gameplay video

- Submit a video at least 7 seconds long demonstrating the game running, the player jumping, and obstacles appearing.

13. Grading Rubric

100 Points

Project assessment

Player movement and jumping	15
Scrolling background	15
Random obstacle spawning	15
Increasing game speed and difficulty	15
Collision detection and Game Over	10
Score display and restart feature	10
Game functionality and testing	10
Project naming, screenshot, and video submission	10

14. Student Reflection Questions

Answer the following questions after completing your game:

1. How does moving the background create the illusion that the player is running?
2. What happens to the game when `game_speed` increases?
3. Why should obstacles appear at random intervals instead of at exactly the same interval?
4. How does gravity affect your player's jumping movement?
5. How does collision detection determine when the game ends?
6. What changes would you make to improve the gameplay experience?

Final objective: Create a fully functional Endless Runner game in which the player jumps over randomly generated obstacles, survives as long as possible, and earns points while the game's speed and difficulty gradually increase.