Pac-Man Exercise in Godot 4.5

In this exercise you will complete a simplified Pac-Man game in Godot for the **"Programming Languages and Paradigms"** seminar at **UZH** (2025). Because the seminar focuses on **GDScript**, the basic project (*level, collisions, animated sprites, and UI*) has already been set up for you.

What you need

Download the Godot Engine from the official website or via Steam. After installation, open Godot, click **Import**, and select the project folder containing `project.godot`.

(*Optional*) GitHub

This exercise is also available on GitHub. You can **fork** the repository and work on your own copy. When you are done, you can create a Pull Request so I can merge it into a solution branch. If you get completely stuck, feel free to open a **GitHub issue**.

Tasks

To keep things beginner-friendly, we will simplify several aspects of the original Pac-Man and focus on what matters for the scripting part. A project template has already been created for you. Your job is to **extend** it with more GDScript code so that it meets the acceptance criteria below. Hints can be found below and inside the GDScript files directly. You do not need to use the template provided.

What you will learn

Although the Godot Engine itself is not the main focus of this assignment, you will inevitably learn your way around a 2D Godot project. You will in particular:

- Use the **node-based scene tree**: each node is a class instance that you can extend with GDScript.
- Work with **`TileMapLayer`** to:
  - read the level layout from tiles,
  - check where walls are,
  - convert between local/screen positions and tile coordinates using functions such as `local_to_map()` and `map_to_local()`,
  - use the tile grid to make Pac-Man turn only at intersections.
- Connect and react to **signals**, for example, to detect when Pac-Man touches a pellet or a ghost.

Hints: Functions and concepts that may help:

- `local_to_map()` / `map_to_local()` on `TileMapLayer`
- `signal` / `connect()` / `await` for timed behavior
- `Input.is_action_pressed()` and `Input.get_vector()`
- `play()` on `AnimatedSprite2D`
- Timers or random intervals via `get_tree().create_timer(...)`

You may also find use in the documentation or the 2d Tutorial.

Acceptance Criteria

The following criteria must be met for the solution to be accepted:

Character movement

- Pac-Man and the Ghosts move on the **x/y tile grid**. Pac-Man is controlled using the **arrow keys** (via `get_input()` in `pac-man.gd`).
- Implement **input buffering**: if the player presses a new direction that is currently blocked, Pac-Man should switch to it as soon as the next intersection allows it.
- A **180° turn** is always allowed immediately.

Warp tunnel

- Pac-Man can leave the maze on the left and appear on the right (*and the other way around*).

Pellets

- When Pac-Man collides with a **pellet**, the pellet is removed from the level.

Animations

- Pac-Man and the ghosts play the correct animation when moving up, down, left, right.

Ghost normal behavior

- All 4 ghosts move around the level **randomly** in their normal mode.

Ghost chase behavior

- All 4 ghosts sometimes try to **chase** the player by navigating towards Pac-Man’s current position.
- This chase behavior is triggered at **random intervals** between 5–15 seconds and lasts 5 seconds.

Win condition

- When **every pellet** has been collected, Pac-Man wins and the game over screen is called with `on_won()` in `game.gd`.

Lose condition

- When Pac-Man **collides** with a ghost, Pac-Man loses and the game over screen is called with `on_lost()` in `game.gd`.

Credits

- Sprites: "General Sprites" by Superjustinbros
- Font: "VR323" by Peter Hull

Contact Information

Gianluca Imbiscuso (gianluca.imbiscuso@uzh.ch)