# Assignment 2: Flappy Bird

## Lecture Update

https://youtu.be/nNZrtKmhtLc

## Objectives

- Read and understand all of the Flappy Bird source code from Lecture 2.
- Create a `PauseState` class.
- Implement the pattern Strategy in order to have a normal and a hard game mode. You can follow this guide.

## Getting Started

- Unzip flappy_bird.zip, which should yield a directory called flappy_bird.
- Enter to the directory, for instance: `cd flappy_bird`
- Compile the source code: `make main`
- Run the game: `./main`

## Specification

1. Create a PauseState in order to stop the game execution. Choose what key you want in order to change from playing to pause state and use the same key to return to playing state. Take into account that you need to keep the world and the bird state between those changes. The method `StateMachine::change_state` receives two optional arguments, they are a pointer to the world and a pointer to the bird. The playing state expects to receive a pointer to the world, if you don't pass the pointer to the bird, a new bird will be created. When you implement the new state, you need to ask yourself the following questions:

    a. Should the state implement the method `enter`?
    b. Should the state implement the method `exit`?
    c. Should the state implement the method `handle_input`?
    d. Should the state implement the method `update`?
    e. Should the state implement the method `render`?
2. By using the Strategy pattern, implement two game modes: Normal and Hard. The normal mode should keep the behavior of the current game. The hard mode should implement the following features:
    a. Allow to move the bird horizontally by using any two keys
    b. Logs are not generated at the same distance and the gap between them should be different, the height of a new log pair should change (respect to the previous one) depending on the distance between them. Take into account not creating an impossible game play.
    c. Some of them are closing and opening the gap. Play a sound when they collide.
    d. Powerup that allows the bird to pass through logs. Generate the powerup in the scene at a random position, use an image to represent it, when the bird takes the powerup, it should be removed from the scene (you can use the Factory pattern to generate it and remove it). The effect of the powerup should have a time duration, the bird should look like a ghost and the music should change. When the effect ends, the bird and the music should come back to the original.

# How to Submit

1. Record a screencast, not to exceed 5 minutes in length (and not uploaded more than two weeks prior to your submission of this project) in which you demonstrate your app's functionality. Upload that video to ouTube (as unlisted or public, but not private) or somewhere else.
2. Upload the source code to a Git repository.
3. Submit this form.

# Deadline

You have to submit this assignment before 02/11/2023 - 23:59:59.